



ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αυτοματοποίηση Μαζικής Διαχείρισης Υπολογιστικών Συστημάτων

Φοιτητής:

Κωνσταντίνος Παπαγεωργίου
Αριθμός Μητρώου: 164718

Επιβλέπων:

Αντώνης Σιδηρόπουλος

1 Σεπτεμβρίου 2026

Title of Dissertation Automation of large-scale computer system management

Code of Dissertation 25200

Student's full name Konstantinos Papageorgiou

Supervisor's full name Antonis Sidiropoulos

Date of undertaking 22-03-2025

Date of completion 01-09-2026

We hereby affirm the authorship of this paper as well as the acknowledgement and credit of whichever assistance We received in its composition. We have, furthermore, noted the various sources from which We extracted data, ideas, visual or written material, in paraphrase or exact quotation. Moreover, we affirm the exclusive composition of this paper by myself only, for the purpose of it being a dissertation, in the Department of Information and Electronic Engineering of the I.H.U.

This paper constitutes the intellectual property of Konstantinos Papageorgiou the student that composed it. According to the open-access policy, the author/composer offers the International Hellenic University authorisation to use the right to reproduce, borrow, publicly present and digitally distribute the paper globally, in electronic form and media of all kinds, for teaching or research purposes, voluntarily. Open access to the full text, by no means grants the right to trespass the intellectual property of the author/composer, nor does it authorise the reproduction, republication, duplication, selling, commercial use, distribution, publication, downloading, uploading, translation, modification of any kind, in part or summary of the paper, without the explicit written consent of the authors.

The approval of this dissertation by the Department of Information and Electronic Engineering of the International Hellenic University, does not necessarily entail the adoption of the author's views, on behalf of the Department.

Πρόλογος

Η διαχείριση μεγάλου αριθμού υπολογιστικών συστημάτων αποτελεί ένα επαναλαμβανόμενο πρόβλημα σε περιβάλλοντα όπου απαιτείται η διατήρηση κοινών ρυθμίσεων, η εγκατάσταση και ενημέρωση λογισμικού, η αντιμετώπιση ανεπιθύμητων μεταβολών και η γρήγορη αποκατάσταση των συστημάτων σε λειτουργική κατάσταση. Όσο αυξάνεται ο αριθμός των διαχειριζόμενων σταθμών, τόσο περισσότερο οι διαδικασίες αυτές παύουν να αποτελούν μεμονωμένες εργασίες συντήρησης και μετατρέπονται σε πρόβλημα συντονισμού και αυτοματοποίησης. Η εκτέλεση των ίδιων ενεργειών ξεχωριστά σε κάθε σύστημα αυξάνει τον απαιτούμενο χρόνο, δημιουργεί περιθώρια ασυνέπειας και καθιστά δυσκολότερη τη διατήρηση ενός προβλέψιμου υπολογιστικού περιβάλλοντος.

Η εξέλιξη των τεχνολογιών εικονικοποίησης προσφέρει τη δυνατότητα διαφορετικής προσέγγισης αυτού του προβλήματος. Η λειτουργία ενός υπολογιστικού περιβάλλοντος δεν χρειάζεται πλέον να συνδέεται αποκλειστικά με έναν συγκεκριμένο φυσικό σταθμό εργασίας, αλλά μπορεί να μεταφερθεί σε κεντρικά διαχειριζόμενες εικονικές μηχανές. Παράλληλα, λειτουργίες όπως η δημιουργία, η διάθεση, η επαναφορά και η αντικατάσταση συστημάτων μπορούν να ενσωματωθούν σε αυτοματοποιημένες διαδικασίες. Η μετάβαση αυτή μεταφέρει το ενδιαφέρον από τη μεμονωμένη διαχείριση κάθε υπολογιστή προς τη διαχείριση μιας κοινής υποδομής και του συνολικού κύκλου ζωής των διαθέσιμων υπολογιστικών πόρων.

Ένα χαρακτηριστικό πεδίο στο οποίο εμφανίζονται αυτές οι ανάγκες είναι τα εκπαιδευτικά εργαστήρια πληροφορικής. Η υποστήριξη πολλών σταθμών εργασίας, διαφορετικών λειτουργικών περιβαλλόντων και διαδοχικών χρηστών δημιουργεί αυξημένες απαιτήσεις για ομοιομορφία, επαναληψιμότητα και γρήγορη επαναφορά. Το συγκεκριμένο περιβάλλον αποτέλεσε την πρακτική αφετηρία της παρούσας εργασίας και οδήγησε στη διερεύνηση μιας προσέγγισης βασισμένης σε υποδομή εικονικών επιφανειών εργασίας και σε μηχανισμούς αυτοματοποίησης της διαχείρισής της.

Στο πλαίσιο αυτό, η εργασία ακολουθεί την πορεία από τη μελέτη του θεωρητικού υποβάθρου και των διαθέσιμων τεχνολογιών έως τον σχεδιασμό, την υλοποίηση και την πειραματική επαλήθευση ενός ολοκληρωμένου συστήματος. Στόχος δεν είναι μόνο η παροχή απομακρυσμένων περιβαλλόντων εργασίας, αλλά κυρίως η διερεύνηση του τρόπου με τον οποίο η εικονικοποίηση, η κεντρική διαχείριση και ο αυτοματισμός μπορούν να συνδυαστούν για τη μείωση επαναλαμβανόμενων διαχειριστικών ενεργειών και τη δημιουργία ενός περισσότερο ελεγχόμενου και επαναχρησιμοποιήσιμου υπολογιστικού περιβάλλοντος.

Abstract

This thesis investigates the automation of large-scale computer system management in educational laboratories, where maintaining numerous physical workstations introduces significant requirements for administration, restoration, and consistency of the available computing environment. To address this problem, a Virtual Desktop Infrastructure (VDI) was designed and implemented using open-source software and the Proxmox VE platform.

The implementation includes Windows and Linux virtual machine templates, separate virtual machine pools, a broker-based mechanism for allocating available systems, and a simplified thin-client environment that allows users to select an operating system and establish remote connections through RDP/XRDP. In parallel, mechanisms were developed for the automated creation, restoration, monitoring, destruction, and replenishment of virtual machines, using snapshots and active-session checks to support safe lifecycle management.

Experimental verification confirmed the correct operation of the complete workflow, from virtual machine creation and allocation to remote access, safe restoration to a known clean state, and subsequent reuse. Concurrent allocation of different virtual machines, protection of active sessions, and automatic restoration of the configured pool capacity were also verified. The results indicate that the proposed approach can provide a functional foundation for the centralized and automated management of educational computer laboratories, within the limitations of the experimental environment evaluated in this work.

Περίληψη

Η παρούσα διπλωματική εργασία εξετάζει την αυτοματοποίηση της μαζικής διαχείρισης υπολογιστικών συστημάτων σε εκπαιδευτικά εργαστήρια, όπου η συντήρηση μεγάλου αριθμού φυσικών σταθμών εργασίας δημιουργεί αυξημένες απαιτήσεις διαχείρισης, επαναφοράς και ομοιομορφίας του διαθέσιμου περιβάλλοντος. Για την αντιμετώπιση του προβλήματος σχεδιάστηκε και υλοποιήθηκε μια υποδομή εικονικών επιφανειών εργασίας (Virtual Desktop Infrastructure – VDI), βασισμένη σε λογισμικό ανοικτού κώδικα και στην πλατφόρμα Proxmox VE.

Η υλοποίηση περιλαμβάνει πρότυπα εικονικών μηχανών Windows και Linux, διακριτές δεξαμενές εικονικών μηχανών, μηχανισμό ανάθεσης διαθέσιμου συστήματος μέσω μεσολαβητή και απλοποιημένο thin-client περιβάλλον για την επιλογή λειτουργικού συστήματος και την απομακρυσμένη σύνδεση μέσω RDP/XRDP. Παράλληλα, αναπτύχθηκαν μηχανισμοί αυτοματοποιημένης δημιουργίας, επαναφοράς, παρακολούθησης, καταστροφής και αναπλήρωσης των εικονικών μηχανών, με χρήση στιγμιοτύπων και ελέγχων ενεργών συνεδριών.

Η πειραματική επαλήθευση επιβεβαίωσε την ορθή λειτουργία της πλήρους αλυσίδας, από τη δημιουργία και ανάθεση μιας εικονικής μηχανής έως την απομακρυσμένη πρόσβαση, την ασφαλή επαναφορά της σε γνωστή καθαρή κατάσταση και την επαναχρησιμοποίησή της. Επιπλέον, επαληθεύτηκαν η παράλληλη ανάθεση διαφορετικών εικονικών μηχανών, η προστασία ενεργών συνεδριών και η αυτόματη αποκατάσταση της επιθυμητής χωρητικότητας των δεξαμενών. Τα αποτελέσματα δείχνουν ότι η προτεινόμενη προσέγγιση μπορεί να αποτελέσει λειτουργική βάση για την κεντρικοποίηση και αυτοματοποίηση της διαχείρισης εκπαιδευτικών εργαστηρίων, εντός των περιορισμών του πειραματικού περιβάλλοντος που αξιολογήθηκε.

Περιεχόμενα

1	Εισαγωγή	1
1.1	Αντικείμενο της Εργασίας	1
1.2	Πλαίσιο Εφαρμογής και Κίνητρο	2
1.3	Στόχος και Προτεινόμενη Προσέγγιση	3
1.4	Συνεισφορά και Πεδίο της Εργασίας	3
1.5	Δομή της Εργασίας	4
2	Θεωρητικό Υπόβαθρο	6
2.1	Εικονικοποίηση Υπολογιστικών Συστημάτων	6
2.1.1	Έννοια της Εικονικοποίησης	6
2.1.2	Εικονικές Μηχανές	6
2.1.3	Hypervisors Τύπου 1 και Τύπου 2	7
2.1.4	Εικονικοποίηση με Υποστήριξη Υλικού	8
2.2	Εικονικοποίηση Επιφάνειας Εργασίας	9
2.2.1	Έννοια της Εικονικοποίησης Επιφάνειας Εργασίας	9
2.2.2	Virtual Desktop Infrastructure	9
2.2.3	Βασική Αρχιτεκτονική VDI	10
2.2.4	Μόνιμες και Μη Μόνιμες Εικονικές Επιφάνειες Εργασίας	10
2.3	Κεντροκοποιημένη Διαχείριση Συστημάτων	12
2.3.1	Παραδοσιακή Διαχείριση Σταθμών Εργασίας	12
2.3.2	Κεντροκοποιημένη Διαχείριση	12
2.3.3	Αυτοματοποίηση Διαχείρισης	13
2.3.4	Πλεονεκτήματα και Περιορισμοί της Κεντροκοποιημένης Διαχείρισης	13
2.4	Απομακρυσμένη Πρόσβαση σε Επιφάνεια Εργασίας	14
2.4.1	Έννοια της Απομακρυσμένης Πρόσβασης	14
2.4.2	Remote Desktop Protocol	14
2.4.3	Εναλλακτικά Πρωτόκολλα Απομακρυσμένης Πρόσβασης	15
2.5	Thin Clients	16
2.5.1	Έννοια και Χαρακτηριστικά	16
2.5.2	Πλεονεκτήματα και Περιορισμοί	16
2.5.3	Thin Clients σε Εκπαιδευτικά Εργαστήρια	17
2.6	Σύνοψη Κεφαλαίου	17
3	Τεχνολογίες και Εργαλεία	19
3.1	Εισαγωγή	19

3.2	Proxmox Virtual Environment	20
3.2.1	Γενική Παρουσίαση	20
3.2.2	Αρχιτεκτονική και Βασικά Χαρακτηριστικά	20
3.2.3	QEMU/KVM και Εικονικές Μηχανές	21
3.2.4	Διαχείριση Εικονικών Μηχανών μέσω Proxmox VE	21
3.2.5	Λόγοι Επιλογής του Proxmox VE	22
3.3	QEMU Guest Agent	23
3.3.1	Ρόλος του Guest Agent	23
3.3.2	Επικοινωνία Host--Guest	23
3.3.3	Αξιοποίηση στο Σύστημα VDI	24
3.4	Λειτουργικά Συστήματα Εικονικών Σταθμών Εργασίας	24
3.4.1	Microsoft Windows	24
3.4.2	GNU/Linux	25
3.4.3	Πρότυπα Εικονικών Μηχανών και Κλωνοποίηση	25
3.5	Remote Desktop Protocol και FreeRDP	26
3.5.1	RDP στην Πρακτική Υλοποίηση VDI	26
3.5.2	FreeRDP	27
3.5.3	XRDP για Συστήματα Linux	27
3.6	Thin Client	28
3.6.1	Debian GNU/Linux ως Βάση του Thin Client	28
3.6.2	X Window System και Γραφικό Περιβάλλον	28
3.6.3	Αυτοματοποιημένη Εκκίνηση και Σύνδεση	29
3.7	SSH και Απομακρυσμένη Εκτέλεση Εντολών	29
3.7.1	OpenSSH	29
3.7.2	Αυθεντικοποίηση και Περιορισμός Πρόσβασης	30
3.7.3	Forced Commands και Αυτοματοποιημένη Επικοινωνία	30
3.8	Bash και Αυτοματοποίηση	31
3.8.1	Shell Scripting	31
3.8.2	Διαχείριση Ταυτόχρονων Αιτημάτων	31
3.8.3	Αρχεία Ρυθμίσεων και Καταγραφή Συμβάντων	32
3.9	Σύνοψη Κεφαλαίου	32
4	Σχεδίαση και Αρχιτεκτονική του Συστήματος	34
4.1	Εισαγωγή	34
4.2	Απαιτήσεις και Στόχοι Σχεδίασης	35
4.3	Συνολική Αρχιτεκτονική του Συστήματος	36
4.4	Αρχιτεκτονική Εικονικών Σταθμών Εργασίας και Pools	37
4.4.1	Πρότυπα και δημιουργία εικονικών σταθμών	37
4.4.2	Διαχωρισμός Linux και Windows pools	38
4.4.3	Κατάσταση ετοιμότητας και σημείο επαναφοράς	39
4.4.4	Διαφοροποίηση της διαδικασίας προετοιμασίας	39
4.4.5	Λογική χωρητικότητας των pools	40
4.5	Σχεδίαση του Broker και Μηχανισμός Ανάθεσης	40
4.5.1	Επιλογή pool και διαθέσιμου μέλους	40
4.5.2	Συγχρονισμός ταυτόχρονων λειτουργιών	41

4.5.3	Εκκίνηση και ανακάλυψη διεύθυνσης δικτύου	41
4.5.4	Διαχωρισμός ανάθεσης και απομακρυσμένης συνεδρίας	42
4.5.5	Περιορισμένη διεπαφή προς τον thin client	42
4.6	Σχεδίαση του Thin Client και Ροή Πρόσβασης Χρήστη	42
4.6.1	Ροή πρόσβασης	43
4.6.2	Ενοποιημένη πρόσβαση σε Windows και Linux	43
4.6.3	Περιορισμός της τοπικής κατάστασης	44
4.7	Σχεδίαση Διαχείρισης Κύκλου Ζωής Εικονικών Μηχανών	44
4.7.1	Δημιουργία και αναπλήρωση των pools	44
4.7.2	Ασφαλής επαναφορά σε καθαρή κατάσταση	45
4.7.3	Καταστροφή και αναπλήρωση διαχειριζόμενων μελών	46
4.7.4	Κοινός συγχρονισμός λειτουργιών	46
4.7.5	Παρακολούθηση της κατάστασης	47
4.8	Ασφάλεια και Διαχωρισμός Δικαιωμάτων	47
4.8.1	Περιορισμένη διαδρομή πρόσβασης	47
4.8.2	Διαχωρισμός ρόλων και δικαιωμάτων	48
4.8.3	Προστασία διαχειριστικών λειτουργιών	48
4.8.4	Διαχωρισμός επιπέδου ελέγχου και συνεδρίας χρήστη	49
4.8.5	Περιορισμοί του μοντέλου ασφάλειας	49
4.9	Όρια Σχεδίασης και Δυνατότητες Επέκτασης	50
4.9.1	Όρια της παρούσας σχεδίασης	50
4.9.2	Δυνατότητες επέκτασης	51
4.10	Σύνοψη Κεφαλαίου	52
5	Υλοποίηση του Συστήματος	53
5.1	Εισαγωγή	53
5.2	Πειραματικό Περιβάλλον και Υποδομή	53
5.3	Προετοιμασία των Golden Images και Templates	55
5.3.1	Προετοιμασία του Windows Golden Image	55
5.3.2	Προετοιμασία του Linux Golden Image	56
5.3.3	Δημιουργία των Τελικών Templates και Χρήση Linked Clones	57
5.4	Υλοποίηση του Broker	57
5.4.1	Περιορισμένο SSH Entry Point και Διαχωρισμός Δικαιωμάτων	58
5.4.2	Επιλογή Pool και Ανάγνωση Διαθέσιμων VMIDs	59
5.4.3	Επιλογή και Εκκίνηση Διαθέσιμης Εικονικής Μηχανής	60
5.4.4	Ανάκτηση της IPv4 Διεύθυνσης μέσω QEMU Guest Agent	60
5.4.5	Έξοδος προς τον Client, Καταγραφή και Χειρισμός Αποτυχιών	61
5.5	Υλοποίηση του Thin Client	61
5.5.1	Αυτόματη Είσοδος και Επιλογή Λειτουργικού Συστήματος	62
5.5.2	Εκκίνηση της Γραφικής Συνεδρίας και Σύνδεση μέσω FreeRDP	63
5.5.3	Χειρισμός Αποτυχιών και Τοπική Καταγραφή	64
5.6	Υλοποίηση Διαχείρισης Κύκλου Ζωής	64
5.6.1	Κοινή Διαμόρφωση και Pool Files	65
5.6.2	Δημιουργία και Αναπλήρωση Pool με το vdi-create-pool	66
5.6.3	Επαναφορά Καθαρής Κατάστασης με το vdi-reset-pool	67

5.6.4	Ελεγχόμενη Καταστροφή με το vdi-destroy-pool	68
5.6.5	Εποπτεία της Κατάστασης με το vdi-status	70
5.7	Τελική Ολοκλήρωση των Υποσυστημάτων	71
5.7.1	Κοινός Μηχανισμός Κλειδώματος	71
5.7.2	Καταγραφή και Χειρισμός Σφαλμάτων	72
5.7.3	Τελική Ροή Λειτουργίας	73
5.8	Πρακτικά Ζητήματα και Προσαρμογές κατά την Υλοποίηση	74
5.8.1	Λήξη Κωδικού του Κοινού Λογαριασμού Windows	74
5.8.2	Πρόβλημα Sysprep από AppX Package	74
5.8.3	Πρώτη Εκκίνηση των Windows Clones	75
5.8.4	Σειρά Εφαρμογής Tags και Δημιουργίας Snapshot	75
5.8.5	Ανίχνευση Ενεργών RDP Συνεδριών	76
5.8.6	Κοινός Χώρος Locks και Δικαιώματα Πρόσβασης	76
5.8.7	Αυτοματοποίηση της Εκκίνησης του Thin Client	77
5.9	Σύνοψη Κεφαλαίου	77
6	Δοκιμές, Επαλήθευση και Αξιολόγηση του Συστήματος	79
6.1	Εισαγωγή	79
6.2	Έλεγχος Δημιουργίας και Κατάστασης των VDI Pools	79
6.3	Έλεγχος Ανάθεσης και Απομακρυσμένης Πρόσβασης	80
6.3.1	Έλεγχος Windows Pool	80
6.3.2	Έλεγχος Linux Pool	81
6.3.3	Αξιολόγηση της End-to-End Ροής	81
6.4	Έλεγχος Ταυτόχρονων Αιτημάτων Ανάθεσης	81
6.5	Έλεγχος Επαναφοράς και Προστασίας Ενεργών Συνεδριών	82
6.5.1	Έλεγχος Περιόδου Προστασίας	82
6.5.2	Έλεγχος Ενεργής Συνεδρίας	83
6.5.3	Rollback και Επαναχρησιμοποίηση	83
6.5.4	Έλεγχος Επαναφοράς Ολόκληρου Pool	83
6.6	Έλεγχος Καταστροφής και Αναπλήρωσης του Pool	84
6.6.1	Καταστροφή Ολόκληρου του Pool	84
6.6.2	Αυτόματη Αναπλήρωση της Χωρητικότητας	84
6.6.3	Αξιολόγηση του Κύκλου Καταστροφής και Αναπλήρωσης	85
6.7	Έλεγχος Παρακολούθησης της Κατάστασης	85
6.8	Αξιολόγηση των Αποτελεσμάτων	86
6.8.1	Λειτουργική Κάλυψη της Υλοποίησης	86
6.8.2	Αξιολόγηση των Μηχανισμών Προστασίας	86
6.8.3	Όρια της Πειραματικής Επαλήθευσης	86
6.9	Σύνοψη Κεφαλαίου	87
7	Συμπεράσματα και Μελλοντικές Επεκτάσεις	89
7.1	Εισαγωγή	89
7.2	Συνολική Αποτίμηση της Εργασίας	89
7.3	Βασικά Αποτελέσματα και Συνεισφορά	90
7.4	Περιορισμοί της Παρούσας Υλοποίησης	91

7.5	Μελλοντικές Επεκτάσεις	92
7.6	Τελικό Συμπέρασμα	93
Βιβλιογραφία		95

Κατάλογος σχημάτων

2.1	Γενική αρχιτεκτονική εικονικοποίησης υπολογιστικών συστημάτων και θέση του hypervisor στις αρχιτεκτονικές Type 1 και Type 2.	8
4.1	Συνολική αρχιτεκτονική του προτεινόμενου συστήματος VDI.	38
5.1	Λογική ελέγχου και ασφαλούς επαναφοράς εικονικής μηχανής.	69

Κατάλογος πινάκων

2.1	Σύγκριση μόνιμων και μη μόνιμων εικονικών επιφανειών εργασίας.	11
5.1	Βασικές παράμετροι του πειραματικού περιβάλλοντος	54
6.1	Τελική κατάσταση των VDI pools μετά τη δημιουργία	80
6.2	Σύνοψη των βασικών λειτουργικών ελέγχων	88

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της Εργασίας

Η διαχείριση υπολογιστικών συστημάτων αποτελεί βασική λειτουργία κάθε οργανωμένης πληροφοριακής υποδομής. Καθώς ο αριθμός των συστημάτων αυξάνεται, εργασίες όπως η εγκατάσταση και ενημέρωση λογισμικού, η εφαρμογή κοινών ρυθμίσεων, η παρακολούθηση της κατάστασης των συστημάτων και η αποκατάστασή τους μετά από ανεπιθύμητες μεταβολές αποκτούν σημαντικό διαχειριστικό κόστος. Ενέργειες που είναι σχετικά απλές όταν αφορούν έναν μεμονωμένο υπολογιστή μπορούν να εξελιχθούν σε χρονοβόρες και επαναλαμβανόμενες διαδικασίες όταν πρέπει να εφαρμοστούν με συνέπεια σε δεκάδες ή περισσότερα συστήματα. Το πρόβλημα δεν αφορά μόνο τον συνολικό χρόνο που απαιτείται για την εκτέλεση αυτών των εργασιών. Η ανεξάρτητη διαχείριση μεγάλου αριθμού υπολογιστών δυσχεραίνει και τη διατήρηση μιας ομοιόμορφης κατάστασης μεταξύ τους. Διαφορετικές εκδόσεις λογισμικού, αποκλίσεις στις ρυθμίσεις ή μεταβολές που έχουν πραγματοποιηθεί σε συγκεκριμένα συστήματα μπορούν σταδιακά να οδηγήσουν σε ένα ανομοιογενές σύνολο σταθμών, ακόμη και όταν αυτοί προορίζονται να εξυπηρετούν τον ίδιο σκοπό. Κατά συνέπεια, όσο αυξάνεται το μέγεθος της υποδομής, τόσο μεγαλύτερη σημασία αποκτά η δυνατότητα εφαρμογής κοινών πολιτικών και διαδικασιών με προβλέψιμο και επαναλαμβανόμενο τρόπο.

Η κεντρικοποίηση και η αυτοματοποίηση αποτελούν δύο βασικές κατευθύνσεις για την αντιμετώπιση του προβλήματος αυτού. Αντί κάθε υπολογιστής να αντιμετωπίζεται αποκλειστικά ως ανεξάρτητη μονάδα, ένα σύνολο συστημάτων μπορεί να οργανωθεί και να διαχειρίζεται μέσω κοινών μηχανισμών. Επαναλαμβανόμενες λειτουργίες μπορούν αντίστοιχα να μετατραπούν σε αυτοματοποιημένες διαδικασίες, μειώνοντας την ανάγκη για χειροκίνητη παρέμβαση και διευκολύνοντας τη συνεπή εφαρμογή των ίδιων ενεργειών σε πολλαπλά συστήματα. Το ζητούμενο της μαζικής διαχείρισης συνεπώς είναι η αντιμετώπιση των υπολογιστών ως τμήματα μιας ενιαίας και ελεγχόμενης υποδομής.

Οι ανάγκες αυτές εμφανίζονται σε διαφορετικά περιβάλλοντα στα οποία χρησιμοποιείται μεγάλος αριθμός υπολογιστικών συστημάτων. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι περιπτώσεις όπου πολλοί σταθμοί καλούνται να παρέχουν παρόμοια ή προκαθορισμένα περιβάλλοντα εργασίας και όπου η κατάσταση των συστημάτων πρέπει να παραμένει ελεγχόμενη μεταξύ διαδοχικών χρήσεων. Σε τέτοιες περιπτώσεις, η δυνατότητα κεντρικής προετοιμασίας, διάθεσης και αποκατάστασης των υπολογιστικών περιβαλλόντων μπορεί να περιορίσει σημαντικά τις επαναλαμβανόμενες εργασίες διαχείρισης.

Χαρακτηριστικό παράδειγμα αποτελούν τα εκπαιδευτικά εργαστήρια υπολογιστών. Σε αυτά συνυπάρχουν πολλοί σταθμοί εργασίας, διαδοχικοί χρήστες και διαφορετικές απαιτήσεις που προκύπτουν από τα μαθήματα και τις υπόλοιπες εκπαιδευτικές δραστηριότητες. Παράλληλα, είναι συχνά επιθυμητό οι διαθέσιμοι σταθμοί να παρουσιάζουν ομοιογενή και προβλέψιμη συμπεριφορά και να μπορούν να προετοιμάζονται ή να επανέρχονται σε συγκεκριμένη κατάσταση χωρίς εκτεταμένη παρέμβαση σε κάθε φυσικό υπολογιστή ξεχωριστά.

Με βάση το παραπάνω πρόβλημα, η παρούσα εργασία εξετάζει την αυτοματοποίηση της μαζικής διαχείρισης υπολογιστικών συστημάτων με πεδίο εφαρμογής ένα εκπαιδευτικό εργαστήριο. Η διερεύνηση επικεντρώνεται στη δυνατότητα μεταφοράς σημαντικού μέρους της διαχείρισης από τους επιμέρους φυσικούς σταθμούς σε κεντρικά ελεγχόμενη υποδομή, έτσι ώστε η προετοιμασία, η διάθεση και η επαναφορά των περιβαλλόντων εργασίας να μπορούν να πραγματοποιούνται με ενιαίο και σε σημαντικό βαθμό αυτοματοποιημένο τρόπο. Οι ιδιαίτερες συνθήκες που καθιστούν τα εκπαιδευτικά εργαστήρια κατάλληλο πεδίο εφαρμογής της προσέγγισης αυτής παρουσιάζονται στην επόμενη ενότητα.

1.2 Πλαίσιο Εφαρμογής και Κίνητρο

Τα εκπαιδευτικά εργαστήρια υπολογιστών αποτελούν ένα χαρακτηριστικό περιβάλλον στο οποίο οι απαιτήσεις μαζικής διαχείρισης εμφανίζονται με ιδιαίτερη ένταση. Ένα εργαστήριο περιλαμβάνει συνήθως πολλούς σταθμούς εργασίας οι οποίοι εξυπηρετούν κοινό ή παρόμοιο σκοπό, χωρίς όμως να αντιστοιχούν μόνιμα σε συγκεκριμένους χρήστες. Οι ίδιοι υπολογιστές χρησιμοποιούνται διαδοχικά από διαφορετικούς φοιτητές και στο πλαίσιο διαφορετικών μαθημάτων, γεγονός που δημιουργεί ανάγκη για συνεπή και ελεγχόμενη διαμόρφωση των θέσεων συστημάτων.

Οι απαιτήσεις του εργαστηρίου μπορούν παράλληλα να μεταβάλλονται ανάλογα με την εκπαιδευτική δραστηριότητα. Διαφορετικά μαθήματα ενδέχεται να απαιτούν διαφορετικές εφαρμογές, εργαλεία ανάπτυξης, ρυθμίσεις ή λειτουργικά συστήματα. Η προετοιμασία και διατήρηση αυτών των περιβαλλόντων ανεξάρτητα σε κάθε φυσικό σταθμό αυξάνει τον διαχειριστικό φόρτο και δημιουργεί την ανάγκη εφαρμογής των ίδιων αλλαγών με συνέπεια σε μεγάλο αριθμό υπολογιστών.

Εξίσου σημαντική είναι η κατάσταση στην οποία παραμένει ένας σταθμός μετά τη χρήση του. Σε ένα κοινόχρηστο εκπαιδευτικό περιβάλλον, αλλαγές στις ρυθμίσεις, αρχεία ή άλλες παρεμβάσεις μιας συνεδρίας μπορούν να επηρεάσουν την επόμενη. Αυτό αποκτά ιδιαίτερη σημασία σε περιπτώσεις όπου απαιτείται όλοι οι χρήστες να ξεκινούν από ένα κοινό και προβλέψιμο περιβάλλον, όπως κατά τη διεξαγωγή οργανωμένων εργαστηριακών ασκήσεων ή εξετάσεων. Η δυνατότητα επαναφοράς των συστημάτων σε μία γνωστή αρχική κατάσταση αποτελεί επομένως σημαντική λειτουργική απαίτηση.

Τα χαρακτηριστικά αυτά καθιστούν το εκπαιδευτικό εργαστήριο κατάλληλο πεδίο για τη διερεύνηση μεθόδων κεντρικής και αυτοματοποιημένης διαχείρισης. Η ύπαρξη πολλών παρόμοιων σταθμών, η επαναλαμβανόμενη χρήση τους, η ανάγκη υποστήριξης διαφορετικών περιβαλλόντων και η απαίτηση επαναφοράς σε προβλέψιμη κατάσταση δημιουργούν ένα σαφές κίνητρο για την αναζήτηση μιας προσέγγισης που μειώνει την εξάρτηση από τη χειροκίνητη διαχείριση κάθε φυσικού υπολογιστή ξεχωριστά. Με βάση αυτό το πλαίσιο, στην επόμενη ενότητα προσδιορίζεται ο στόχος της εργασίας και παρουσιάζεται συνοπτικά η προσέγγιση που

επιλέχθηκε για την αντιμετώπιση του προβλήματος.

1.3 Στόχος και Προτεινόμενη Προσέγγιση

Στόχος της παρούσας εργασίας είναι η σχεδίαση, υλοποίηση και πειραματική επαλήθευση ενός συστήματος για την κεντρική και αυτοματοποιημένη διαχείριση πολλαπλών υπολογιστικών περιβαλλόντων, με πεδίο εφαρμογής ένα εκπαιδευτικό εργαστήριο. Η επιδιωκόμενη λύση πρέπει να περιορίζει τις επαναλαμβανόμενες ενέργειες που απαιτούνται για την προετοιμασία και συντήρηση ανεξάρτητων σταθμών εργασίας και να επιτρέπει τη διαχείριση των διαθέσιμων περιβαλλόντων ως μέρη μιας κοινής υποδομής.

Για την επίτευξη του στόχου αυτού επιλέχθηκε μία προσέγγιση βασισμένη σε υποδομή εικονικών επιφανειών εργασίας (Virtual Desktop Infrastructure -- VDI). Τα περιβάλλοντα εργασίας εκτελούνται ως εικονικές μηχανές σε κεντρική υποδομή εικονικοποίησης, ενώ οι τελικοί σταθμοί χρησιμοποιούνται κυρίως ως σημεία πρόσβασης σε αυτά. Η προσέγγιση αυτή επιτρέπει τον διαχωρισμό του περιβάλλοντος που χρησιμοποιεί ο τελικός χρήστης από τη διαμόρφωση του φυσικού τερματικού και μεταφέρει σημαντικό μέρος των λειτουργιών προετοιμασίας και διαχείρισης σε κεντρικό επίπεδο.

Στο πλαίσιο της εργασίας υλοποιήθηκε ένα πειραματικό σύστημα που υποστηρίζει περιβάλλοντα Windows και Linux, οργανωμένα σε διαχειριζόμενες δεξαμενές εικονικών μηχανών. Η πρόσβαση πραγματοποιείται μέσω απλού thin client, ενώ ένας ενδιάμεσος μηχανισμός κατανομής αναλαμβάνει την επιλογή και προετοιμασία κατάλληλου διαθέσιμου εικονικού σταθμού για κάθε αίτημα. Παράλληλα, αναπτύχθηκαν αυτοματοποιημένες λειτουργίες για τη δημιουργία και διατήρηση των δεξαμενών, την επαναφορά των εικονικών σταθμών σε προκαθορισμένη καθαρή κατάσταση, την ελεγχόμενη κατάργησή τους και την παρακολούθηση της λειτουργικής κατάστασης της υποδομής. Με τον τρόπο αυτό, η διαχείριση δεν περιορίζεται στην απομακρυσμένη πρόσβαση σε εικονικές μηχανές, αλλά καλύπτει βασικά στάδια του κύκλου ζωής τους.

Η προτεινόμενη λύση αναπτύχθηκε ως λειτουργικό πρωτότυπο και αξιολογήθηκε μέσω πειραματικών δοκιμών των βασικών ροών και μηχανισμών αυτοματοποίησης. Σκοπός της αξιολόγησης είναι να εξεταστεί κατά πόσο η σχεδιασμένη προσέγγιση μπορεί να υλοποιήσει με συνεπή τρόπο την κατανομή, χρήση και διαχείριση πολλαπλών εικονικών σταθμών και όχι να προσδιοριστούν ποσοτικά όρια απόδοσης ή κλιμάκωσης μιας παραγωγικής εγκατάστασης. Με βάση το πλαίσιο αυτό, στην επόμενη ενότητα συνοψίζονται η συγκεκριμένη συνεισφορά της εργασίας και τα όρια εντός των οποίων πρέπει να αξιολογηθούν τα αποτελέσματά της.

1.4 Συνεισφορά και Πεδίο της Εργασίας

Η κύρια συνεισφορά της εργασίας βρίσκεται στη σχεδίαση, υλοποίηση και λειτουργική επαλήθευση μιας ολοκληρωμένης διαδικασίας μαζικής διαχείρισης εικονικών σταθμών εργασίας. Η εργασία δεν προτείνει μία νέα τεχνολογία εικονικοποίησης ή απομακρυσμένης πρόσβασης, αλλά συνδυάζει υπάρχουσες τεχνολογίες σε ένα ενιαίο σύστημα προσανατολισμένο στην αυτοματοποίηση της διαχείρισης. Στο σύστημα αυτό εντάσσονται η δημιουργία και οργάνωση εικονικών σταθμών Windows και Linux, η αυτόματη κατανομή τους στους χρήστες, η επα-

ναφορά τους σε γνωστή κατάσταση και οι βασικές λειτουργίες διαχείρισης του κύκλου ζωής τους.

Ιδιαίτερη έμφαση δόθηκε στη σύνδεση των επιμέρους λειτουργιών σε μία συνεπή ροή, ώστε οι εικονικές μηχανές να αντιμετωπίζονται ως μέλη διαχειριζόμενων δεξαμενών και όχι ως ανεξάρτητα συστήματα. Η υλοποίηση περιλαμβάνει μηχανισμούς για τον συντονισμό ταυτόχρονων ενεργειών, την προστασία ενεργών συνεδριών κατά τις διαδικασίες επαναφοράς και τον περιορισμό των διοικητικών ενεργειών στους πόρους που ανήκουν στη διαχειριζόμενη υποδομή. Οι βασικές αυτές λειτουργίες εξετάστηκαν πειραματικά, επιτρέποντας παράλληλα τον εντοπισμό και τη διόρθωση προβλημάτων που εμφανίστηκαν κατά την επαλήθευση της τελικής υλοποίησης.

Η ανάπτυξη πραγματοποιήθηκε με στόχο μία σχετικά απλή και αναπαραγώγιμη λύση, αξιολογώντας ελεύθερο και ανοικτού κώδικα λογισμικό όπου αυτό ήταν εφικτό και χωρίς εξάρτηση από μία εξειδικευμένη εμπορική πλατφόρμα VDI. Παράλληλα, η λύση σχεδιάστηκε ώστε η αλληλεπίδραση του τελικού χρήστη με την υποδομή να παραμένει περιορισμένη, ενώ οι λειτουργίες επιλογής και διαχείρισης των πραγματικών υπολογιστικών πόρων εκτελούνται κεντρικά και αυτοματοποιημένα.

Το πεδίο της εργασίας περιορίζεται στην ανάπτυξη και λειτουργική αξιολόγηση ενός πειραματικού πρωτοτύπου. Η υποδομή υλοποιήθηκε σε ελεγχόμενο εικονικοποιημένο περιβάλλον και δεν αναπτύχθηκε ως εγκατάσταση πλήρους κλίμακας σε φυσικό εκπαιδευτικό εργαστήριο. Οι δοκιμές επικεντρώθηκαν στην ορθότητα των βασικών λειτουργιών και των μηχανισμών αυτοματοποίησης και όχι στη μέτρηση της απόδοσης υπό μεγάλο αριθμό ταυτόχρονων χρηστών. Για τον λόγο αυτό, η εργασία δεν επιχειρεί να προσδιορίσει ποσοτικά τη μέγιστη χωρητικότητα ή την κλιμάκωση μιας αντίστοιχης παραγωγικής εγκατάστασης.

Επιπλέον, δεν πραγματοποιήθηκε συστηματική αξιολόγηση διαφορετικών κατηγοριών φυσικού εξοπλισμού ως *thin clients*, ούτε αναπτύχθηκαν λειτουργίες όπως εξατομικευμένη διαχείριση χρηστών, υψηλή διαθεσιμότητα, ενορχήστρωση πολλαπλών κόμβων ή δυναμική προσαρμογή της χωρητικότητας ανάλογα με τη ζήτηση. Οι δυνατότητες αυτές αποτελούν πιθανές επεκτάσεις μιας πληρέστερης εγκατάστασης, αλλά βρίσκονται εκτός του πειραματικού πεδίου της παρούσας εργασίας.

Κατά συνέπεια, τα αποτελέσματα της εργασίας πρέπει να ερμηνεύονται κυρίως ως επαλήθευση της λειτουργικής εφικτότητας της προτεινόμενης προσέγγισης. Το πρωτότυπο καταδεικνύει ότι οι βασικές διαδικασίες διάθεσης και διαχείρισης πολλαπλών εικονικών σταθμών μπορούν να συνδυαστούν σε μία κεντρικά ελεγχόμενη και σε σημαντικό βαθμό αυτοματοποιημένη ροή. Η αξιολόγηση της ίδιας προσέγγισης σε μεγαλύτερη κλίμακα και σε πραγματική φυσική υποδομή αποτελεί διαφορετικό στάδιο, το οποίο δεν επιχειρείται στο πλαίσιο της παρούσας εργασίας.

ok

1.5 Δομή της Εργασίας

Η εργασία οργανώνεται σε επτά κεφάλαια, ακολουθώντας τη μετάβαση από τη θεωρητική θεμελίωση του προβλήματος προς τη σχεδίαση και υλοποίηση της προτεινόμενης λύσης, την πειραματική επαλήθευσή της και την τελική αξιολόγηση των αποτελεσμάτων.

Το δεύτερο κεφάλαιο παρουσιάζει το θεωρητικό υπόβαθρο της εργασίας, με έμφαση στην ει-

κονικοποίηση υπολογιστικών συστημάτων και επιφάνειας εργασίας, στην κεντριοποιημένη διαχείριση, στην απομακρυσμένη πρόσβαση και στην έννοια των thin clients.

Το τρίτο κεφάλαιο παρουσιάζει τις βασικές τεχνολογίες που αξιοποιούνται στην υλοποίηση και τον ρόλο τους στο σύστημα, καλύπτοντας την πλατφόρμα εικονικοποίησης, τα λειτουργικά περιβάλλοντα, τις τεχνολογίες απομακρυσμένης πρόσβασης και τους μηχανισμούς αυτοματοποίησης και επικοινωνίας.

Το τέταρτο κεφάλαιο επικεντρώνεται στη σχεδίαση και την αρχιτεκτονική του συστήματος. Παρουσιάζονται οι απαιτήσεις που καθόρισαν τη σχεδίαση, τα βασικά δομικά στοιχεία της λύσης και ο τρόπος με τον οποίο οργανώνονται η κατανομή των εικονικών σταθμών και η διαχείριση του κύκλου ζωής τους.

Το πέμπτο κεφάλαιο περιγράφει την υλοποίηση του συστήματος, παρουσιάζοντας το πειραματικό περιβάλλον και τις επιμέρους ρυθμίσεις και μηχανισμούς μέσω των οποίων η σχεδιασμένη αρχιτεκτονική μετατράπηκε σε λειτουργικό πρωτότυπο.

Το έκτο κεφάλαιο παρουσιάζει την πειραματική επαλήθευση της υλοποίησης. Καταγράφονται οι δοκιμές των βασικών λειτουργιών του συστήματος, τα αποτελέσματά τους και τα προβλήματα που εντοπίστηκαν και διορθώθηκαν κατά τη διαδικασία επαλήθευσης.

Τέλος, το έβδομο κεφάλαιο πραγματοποιεί τη συνολική αποτίμηση της εργασίας, συνοψίζει τα βασικά αποτελέσματα και τη συνεισφορά της, εξετάζει τους περιορισμούς της πειραματικής υλοποίησης και παρουσιάζει πιθανές κατευθύνσεις για μελλοντική επέκταση.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Εικονικοποίηση Υπολογιστικών Συστημάτων

2.1.1 Έννοια της Εικονικοποίησης

Η εικονικοποίηση (virtualization) αποτελεί τεχνολογία μέσω της οποίας οι φυσικοί υπολογιστικοί πόροι ενός συστήματος μπορούν να αναπαρασταθούν και να διατεθούν ως λογικοί ή εικονικοί πόροι. Με την εισαγωγή ενός επιπέδου αφαίρεσης (abstraction layer) μεταξύ του φυσικού υλικού και του λογισμικού που το χρησιμοποιεί, πόροι όπως η υπολογιστική ισχύς, η κύρια μνήμη, ο αποθηκευτικός χώρος και οι δικτυακές διεπαφές μπορούν να κατανέμονται μεταξύ περισσότερων ανεξάρτητων υπολογιστικών περιβαλλόντων [1].

Βασική αρχή της εικονικοποίησης είναι επομένως ο διαχωρισμός της λογικής αναπαράστασης ενός υπολογιστικού πόρου από τη φυσική του υλοποίηση. Ένα εικονικοποιημένο περιβάλλον δεν απαιτεί αποκλειστική αντιστοίχιση των πόρων που χρησιμοποιεί σε συγκεκριμένο φυσικό σύστημα. Αντίθετα, οι διαθέσιμοι πόροι μπορούν να συγκεντρώνονται και να κατανέμονται μεταξύ διαφορετικών εικονικών περιβαλλόντων, επιτρέποντας την ταυτόχρονη και απομονωμένη λειτουργία τους πάνω σε κοινή φυσική υποδομή [2].

Η προσέγγιση αυτή επιτρέπει την αποτελεσματικότερη αξιοποίηση του διαθέσιμου υλικού και παρέχει μεγαλύτερη ευελιξία στη διαχείριση των υπολογιστικών πόρων, καθώς η κατανομή τους μπορεί να μεταβάλλεται σε επίπεδο λογισμικού χωρίς αντίστοιχες αλλαγές στη φυσική υποδομή [1]. Η εικονικοποίηση μπορεί να εφαρμοστεί σε διαφορετικά επίπεδα και να αφορά, μεταξύ άλλων, υπολογιστικά συστήματα, αποθηκευτικούς πόρους, δίκτυα και περιβάλλοντα επιφάνειας εργασίας. Για την παρούσα εργασία ιδιαίτερο ενδιαφέρον παρουσιάζουν η εικονικοποίηση υπολογιστικών συστημάτων και η εικονικοποίηση επιφάνειας εργασίας, καθώς αποτελούν τη βάση για την ανάπτυξη υποδομών Virtual Desktop Infrastructure (VDI).

2.1.2 Εικονικές Μηχανές

Μία εικονική μηχανή (Virtual Machine -- VM) αποτελεί ένα απομονωμένο υπολογιστικό περιβάλλον στο οποίο ένα λειτουργικό σύστημα έχει στη διάθεσή του εικονική αναπαράσταση των πόρων ενός φυσικού υπολογιστή. Οι πόροι αυτοί μπορούν να περιλαμβάνουν επεξεργαστή, κύρια μνήμη, αποθηκευτικά μέσα και δικτυακές διεπαφές. Το λειτουργικό σύστημα που εκτελείται στο εσωτερικό της VM αναφέρεται ως φιλοξενούμενο λειτουργικό σύστημα (guest

operating system), ενώ το σύστημα που παρέχει τους φυσικούς πόρους αποτελεί το σύστημα φιλοξενίας (host).

Η αντιστοίχιση μεταξύ εικονικών και πραγματικών πόρων πραγματοποιείται μέσω του επιπέδου εικονικοποίησης. Με αυτόν τον τρόπο, διαφορετικές εικονικές μηχανές μπορούν να χρησιμοποιούν την ίδια φυσική υποδομή χωρίς κάθε λειτουργικό σύστημα να απαιτεί αποκλειστικό φυσικό υλικό. Παράλληλα, οι πόροι κάθε VM μπορούν να καθορίζονται ανεξάρτητα, ανάλογα με τις ανάγκες του αντίστοιχου περιβάλλοντος.

Βασικό χαρακτηριστικό των εικονικών μηχανών είναι η λογική απομόνωσή τους. Παρόλο που περισσότερες VM μπορούν να εκτελούνται ταυτόχρονα στο ίδιο φυσικό σύστημα, καθεμία αποτελεί ξεχωριστό περιβάλλον εκτέλεσης με δικό της λειτουργικό σύστημα και εφαρμογές. Η ιδιότητα αυτή επιτρέπει τη συνύπαρξη διαφορετικών λειτουργικών συστημάτων και φορτίων εργασίας πάνω σε κοινή φυσική υποδομή [2].

Το υλικό που παρουσιάζεται σε μία VM δεν είναι απαραίτητο να αντιστοιχεί άμεσα σε συγκεκριμένες φυσικές συσκευές. Οι συσκευές μπορούν να εξομοιώνονται από το λογισμικό εικονικοποίησης ή να παρέχονται μέσω παραεικονικοποιημένων (paravirtualized) διεπαφών, σχεδιασμένων για αποτελεσματικότερη επικοινωνία μεταξύ του φιλοξενούμενου συστήματος και του επιπέδου εικονικοποίησης. Χαρακτηριστικό παράδειγμα αποτελούν οι συσκευές VirtIO που χρησιμοποιούνται σε περιβάλλοντα βασισμένα στο KVM [3].

Η δυνατότητα εκτέλεσης πολλαπλών ανεξάρτητων εικονικών μηχανών πάνω σε κοινή υποδομή αποτελεί βασικό μηχανισμό για την ενοποίηση και κεντρική διαχείριση υπολογιστικών πόρων. Στην περίπτωση του VDI, οι επιφάνειες εργασίας μπορούν να παρέχονται μέσω ξεχωριστών εικονικών μηχανών, οι οποίες εκτελούνται και διαχειρίζονται κεντρικά.

2.1.3 Hypervisors Τύπου 1 και Τύπου 2

Ο hypervisor, γνωστός επίσης ως Virtual Machine Monitor (VMM), αποτελεί το επίπεδο λογισμικού που επιτρέπει τη δημιουργία και εκτέλεση εικονικών μηχανών πάνω σε κοινή φυσική υποδομή. Διαχειρίζεται τους διαθέσιμους φυσικούς πόρους και την κατανομή τους μεταξύ των εικονικών μηχανών, διατηρώντας παράλληλα τον διαχωρισμό των διαφορετικών περιβαλλόντων εκτέλεσης [4].

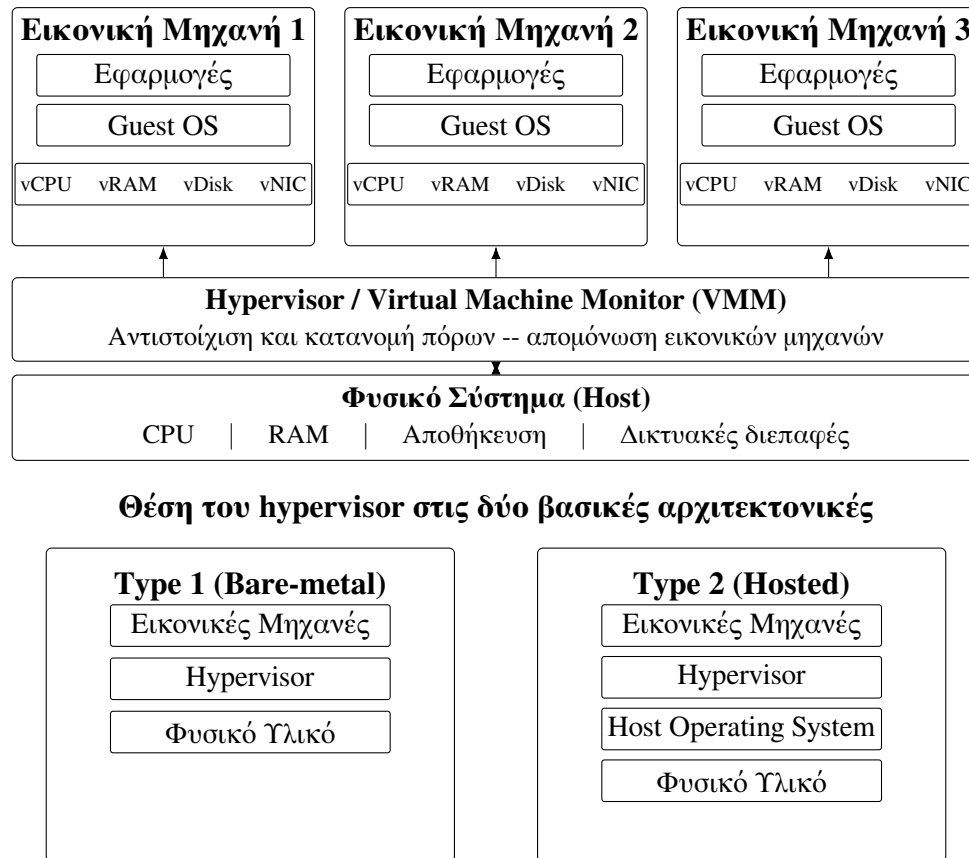
Οι hypervisors διακρίνονται συνήθως σε δύο βασικές κατηγορίες: Τύπου 1 (Type 1) και Τύπου 2 (Type 2). Η διάκριση αφορά τη θέση του επιπέδου εικονικοποίησης ως προς το φυσικό υλικό και το λειτουργικό σύστημα του υπολογιστή [5].

Οι hypervisors Τύπου 1, οι οποίοι αναφέρονται και ως native ή bare-metal hypervisors, λειτουργούν απευθείας πάνω στη φυσική υποδομή και χρησιμοποιούνται ευρέως σε εξυπηρετητές και υποδομές κέντρων δεδομένων, όπου απαιτείται κεντρική εκτέλεση και διαχείριση πολλών εικονικών μηχανών [4]. Αντίθετα, οι hypervisors Τύπου 2, γνωστοί και ως hosted hypervisors, εκτελούνται πάνω σε ένα ήδη εγκατεστημένο λειτουργικό σύστημα και συναντώνται συχνά σε προσωπικούς υπολογιστές και περιβάλλοντα ανάπτυξης ή δοκιμών [4].

Η διάκριση αυτή είναι χρήσιμη για την κατανόηση των διαφορετικών αρχιτεκτονικών εικονικοποίησης. Στην παρούσα εργασία αξιοποιείται το KVM (Kernel-based Virtual Machine), το οποίο παρέχει δυνατότητες εικονικοποίησης μέσω του πυρήνα Linux και αποτελεί βασικό στοιχείο της πλατφόρμας Proxmox VE. Η συγκεκριμένη τεχνολογία και ο ρόλος της στην υλοποίηση εξετάζονται αναλυτικότερα στο επόμενο κεφάλαιο.

Η γενική διαστρωμάτωση ενός εικονικοποιημένου συστήματος, καθώς και η διαφορετική

θέση του hypervisor στις αρχιτεκτονικές Τύπου 1 και Τύπου 2, παρουσιάζονται συνοπτικά στο Σχήμα 2.1.



Σχήμα 2.1: Γενική αρχιτεκτονική εικονικοποίησης υπολογιστικών συστημάτων και θέση του hypervisor στις αρχιτεκτονικές Type 1 και Type 2.

2.1.4 Εικονικοποίηση με Υποστήριξη Υλικού

Η αποτελεσματική εκτέλεση εικονικών μηχανών υποστηρίζεται από δυνατότητες εικονικοποίησης που ενσωματώνονται στους σύγχρονους επεξεργαστές. Οι μηχανισμοί αυτοί επιτρέπουν στον hypervisor να ελέγχει την εκτέλεση των φιλοξενούμενων λειτουργικών συστημάτων με υποστήριξη από το ίδιο το υλικό, περιορίζοντας την ανάγκη εφαρμογής περισσότερο σύνθετων τεχνικών αποκλειστικά μέσω λογισμικού.

Στην αρχιτεκτονική x86, χαρακτηριστικά παραδείγματα αποτελούν οι τεχνολογίες Intel VT-x και AMD-V. Οι επεκτάσεις αυτές παρέχουν μηχανισμούς σε επίπεδο επεξεργαστή για την εκτέλεση και τον έλεγχο εικονικών μηχανών και επιτρέπουν τη λειτουργία συμβατικών φιλοξενούμενων λειτουργικών συστημάτων χωρίς να απαιτείται ειδική τροποποίησή τους για το περιβάλλον εικονικοποίησης [6], [7].

Αντίστοιχη υποστήριξη παρέχεται και για τη διαχείριση της μνήμης. Τεχνολογίες όπως οι Extended Page Tables (EPT) της Intel επιταχύνουν τη μετάφραση μεταξύ της μνήμης που αντιλαμβάνεται το φιλοξενούμενο λειτουργικό σύστημα και της φυσικής μνήμης του host, μειώνον-

ντας την ανάγκη συνεχούς παρέμβασης του hypervisor [6]. Αντίστοιχοι μηχανισμοί παρέχονται και από σύγχρονους επεξεργαστές της AMD[7].

Οι δυνατότητες αυτές αποτελούν σημαντικό τεχνικό υπόβαθρο για σύγχρονες πλατφόρμες ει-
κονικοποίησης και χρησιμοποιούνται από τεχνολογίες όπως το KVM. Για τις ανάγκες της πα-
ρούσας εργασίας δεν απαιτείται περαιτέρω ανάλυση των εσωτερικών μηχανισμών τους, κα-
θώς το ενδιαφέρον επικεντρώνεται στη χρήση της εικονικοποίησης ως βάσης για την κεντρική
παροχή και διαχείριση υπολογιστικών περιβαλλόντων.

2.2 Εικονικοποίηση Επιφάνειας Εργασίας

2.2.1 Έννοια της Εικονικοποίησης Επιφάνειας Εργασίας

Η εικονικοποίηση επιφάνειας εργασίας (desktop virtualization) εφαρμόζει τις αρχές της εικο-
νικοποίησης στο περιβάλλον εργασίας του τελικού χρήστη. Αντί το λειτουργικό σύστημα, οι
εφαρμογές και τα δεδομένα μιας επιφάνειας εργασίας να εξαρτώνται αποκλειστικά από τον
φυσικό υπολογιστή στον οποίο βρίσκεται ο χρήστης, το περιβάλλον μπορεί να εκτελείται σε
διαφορετική υπολογιστική υποδομή και να προσπελαύνεται απομακρυσμένα μέσω δικτύου.

Με τον τρόπο αυτό διαχωρίζεται το περιβάλλον εργασίας από τη φυσική συσκευή πρόσβασης.
Ο τελικός σταθμός αναλαμβάνει κυρίως την αλληλεπίδραση με τον χρήστη και την παρου-
σίαση της απομακρυσμένης επιφάνειας εργασίας, ενώ η εκτέλεση του λειτουργικού συστή-
ματος και των εφαρμογών πραγματοποιείται στην κεντρική υποδομή. Ο διαχωρισμός αυτός
επιτρέπει στο ίδιο περιβάλλον εργασίας να είναι προσβάσιμο από διαφορετικές κατηγορίες
τερματικών, εφόσον υπάρχει κατάλληλη δικτυακή σύνδεση και υποστήριξη του αντίστοιχου
πρωτοκόλλου απομακρυσμένης πρόσβασης [8].

Η εικονικοποίηση επιφάνειας εργασίας δεν ταυτίζεται απαραίτητα με μία συγκεκριμένη αρχι-
τεκτονική υλοποίησης. Το απομακρυσμένο περιβάλλον μπορεί να παρέχεται με διαφορετικούς
τρόπους, ανάλογα με τις απαιτήσεις της υποδομής. Μία από τις σημαντικότερες προσεγγίσεις
είναι η Virtual Desktop Infrastructure (VDI), στην οποία οι επιφάνειες εργασίας εκτελούνται
συνήθως ως εικονικές μηχανές σε κεντρικά διαχειριζόμενη υποδομή.

2.2.2 Virtual Desktop Infrastructure

Ο όρος Virtual Desktop Infrastructure (VDI) αναφέρεται σε αρχιτεκτονική παροχής εικονικών
επιφανειών εργασίας από κεντρική υπολογιστική υποδομή προς απομακρυσμένους χρήστες.
Οι επιφάνειες εργασίας εκτελούνται σε εξυπηρετητές ή συστήματα εικονικοποίησης και η πρό-
σβαση σε αυτές πραγματοποιείται μέσω δικτύου από τους τελικούς σταθμούς [9].

Σε μία τέτοια αρχιτεκτονική, το λειτουργικό σύστημα και οι εφαρμογές της επιφάνειας ερ-
γασίας εκτελούνται στην κεντρική υποδομή και όχι στο τερματικό του χρήστη. Το τερματικό
αποστέλλει κυρίως δεδομένα εισόδου, όπως ενέργειες πληκτρολογίου και ποντικιού, και λαμ-
βάνει την οπτική αναπαράσταση του απομακρυσμένου περιβάλλοντος μέσω κατάλληλου πρω-
τοκόλλου. Κατά συνέπεια, οι απαιτήσεις του τελικού σταθμού μπορούν να διαφοροποιηθούν
σημαντικά από εκείνες ενός υπολογιστή που εκτελεί το πλήρες περιβάλλον εργασίας τοπικά.

Η συγκέντρωση των επιφανειών εργασίας σε κεντρική υποδομή δημιουργεί παράλληλα δυνα-
τότητες κεντρικής διαχείρισης. Η δημιουργία, η διαμόρφωση και η συντήρηση των διαθέσι-

μων περιβαλλόντων μπορούν να πραγματοποιούνται σε επίπεδο υποδομής, αντί να απαιτείται ανεξάρτητη παρέμβαση σε κάθε φυσικό τερματικό. Το χαρακτηριστικό αυτό καθιστά το VDI ιδιαίτερα σχετικό με περιβάλλοντα στα οποία απαιτείται η διάθεση παρόμοιων ή προκαθορισμένων επιφανειών εργασίας σε πολλούς χρήστες.

2.2.3 Βασική Αρχιτεκτονική VDI

Μία τυπική αρχιτεκτονική VDI αποτελείται από περισσότερα συνεργαζόμενα επίπεδα. Στο κεντρικό επίπεδο βρίσκεται η υπολογιστική υποδομή στην οποία εκτελούνται οι εικονικές επιφάνειες εργασίας. Η υποδομή αυτή περιλαμβάνει τους φυσικούς υπολογιστικούς πόρους, το επίπεδο εικονικοποίησης και τις εικονικές μηχανές που παρέχουν τα περιβάλλοντα εργασίας. Στην πλευρά του χρήστη βρίσκονται οι συσκευές πρόσβασης, οι οποίες μπορούν να είναι συμβατικοί προσωπικοί υπολογιστές, φορητές συσκευές ή *thin clients*. Οι συσκευές αυτές χρησιμοποιούν κατάλληλο λογισμικό πελάτη και πρωτόκολλο απομακρυσμένης πρόσβασης για τη σύνδεση με την επιφάνεια εργασίας που εκτελείται στην κεντρική υποδομή.

Μεταξύ των τελικών χρηστών και των διαθέσιμων επιφανειών εργασίας μπορεί να παρεμβάλλεται ένας *connection broker*. Ο *broker* αναλαμβάνει τη διαχείριση των αιτημάτων σύνδεσης και την αντιστοίχισή τους σε κατάλληλο διαθέσιμο περιβάλλον εργασίας. Ανάλογα με την αρχιτεκτονική, η επιλογή αυτή μπορεί να βασίζεται σε στοιχεία όπως η ταυτότητα του χρήστη, η κατάσταση των διαθέσιμων επιφανειών, η συμμετοχή τους σε συγκεκριμένες ομάδες ή άλλες πολιτικές κατανομής [10].

Ο *connection broker* δεν αποτελεί το ίδιο το απομακρυσμένο περιβάλλον εργασίας. Ο ρόλος του είναι κυρίως ο συντονισμός της διαδικασίας μέσω της οποίας ο χρήστης κατευθύνεται προς έναν κατάλληλο πόρο. Μετά την επιλογή του περιβάλλοντος, η πραγματική συνεδρία μπορεί να πραγματοποιείται απευθείας μεταξύ του τερματικού και της αντίστοιχης εικονικής επιφάνειας εργασίας, ανάλογα με τη συγκεκριμένη υλοποίηση.

Η αρχιτεκτονική αυτή διαχωρίζει επομένως τρεις βασικές λειτουργίες: την εκτέλεση των εικονικών επιφανειών εργασίας, την επιλογή ή κατανομή τους και την απομακρυσμένη πρόσβαση από τον τελικό χρήστη. Ο διαχωρισμός αυτός είναι ιδιαίτερα χρήσιμος για την κατανόηση των συστημάτων VDI, καθώς οι αντίστοιχες λειτουργίες μπορούν να υλοποιούνται από διαφορετικά στοιχεία της υποδομής.

2.2.4 Μόνιμες και Μη Μόνιμες Εικονικές Επιφάνειες Εργασίας

Οι εικονικές επιφάνειες εργασίας μπορούν να οργανωθούν με διαφορετικό τρόπο ως προς τη διατήρηση της κατάστασής τους μεταξύ διαδοχικών συνεδριών. Μία βασική διάκριση είναι μεταξύ μόνιμων (*persistent*) και μη μόνιμων (*non-persistent*) επιφανειών εργασίας [11].

Στις μόνιμες επιφάνειες εργασίας, ένας χρήστης συνδέεται συνήθως με συγκεκριμένο εικονικό περιβάλλον, στο οποίο οι αλλαγές μπορούν να διατηρούνται μεταξύ διαφορετικών συνεδριών. Ρυθμίσεις, εγκατεστημένες εφαρμογές και δεδομένα μπορούν επομένως να παραμένουν διαθέσιμα κατά την επόμενη σύνδεση. Η προσέγγιση αυτή προσομοιάζει περισσότερο τη χρήση ενός προσωπικού υπολογιστή και είναι κατάλληλη όταν απαιτείται σημαντικός βαθμός εξατομίκευσης του περιβάλλοντος.

Στις μη μόνιμες επιφάνειες εργασίας, αντίθετα, δεν είναι απαραίτητο ένας χρήστης να αντιστοιχεί μόνιμα στην ίδια εικονική μηχανή. Τα περιβάλλοντα μπορούν να παρέχονται από μία

κοινή ομάδα διαθέσιμων επιφανειών και να επανέρχονται μετά τη χρήση σε μία προκαθορισμένη κατάσταση. Έτσι περιορίζεται η συσσώρευση μεταβολών μεταξύ διαδοχικών συνεδριών και διευκολύνεται η διατήρηση ομοιομορφίας μεταξύ των διαθέσιμων συστημάτων [11].

Η επιλογή μεταξύ των δύο μοντέλων εξαρτάται από τις απαιτήσεις του περιβάλλοντος. Οι μόνιμες επιφάνειες ευνοούν την εξατομίκευση και τη διατήρηση της κατάστασης του χρήστη, αλλά αυξάνουν τις απαιτήσεις διαχείρισης διαφορετικών περιβαλλόντων. Οι μη μόνιμες επιφάνειες περιορίζουν την εξατομίκευση, αλλά είναι ιδιαίτερα κατάλληλες όταν ζητούμενο αποτελεί η διάθεση ομοιόμορφων και επαναχρησιμοποιήσιμων σταθμών εργασίας σε διαφορετικούς χρήστες.

Η δεύτερη περίπτωση παρουσιάζει ιδιαίτερο ενδιαφέρον για κοινόχρηστα περιβάλλοντα, όπως εκπαιδευτικά εργαστήρια, όπου οι ίδιοι υπολογιστικοί πόροι χρησιμοποιούνται διαδοχικά από διαφορετικούς χρήστες και είναι συχνά επιθυμητό κάθε νέα χρήση να ξεκινά από μία γνωστή κατάσταση. Οι συγκεκριμένοι μηχανισμοί με τους οποίους εφαρμόζεται αυτή η αρχή στο σύστημα της παρούσας εργασίας εξετάζονται στα κεφάλαια σχεδίασης και υλοποίησης.

Οι βασικές διαφορές μεταξύ μόνιμων και μη μόνιμων εικονικών επιφανειών εργασίας συνοψίζονται στον Πίνακα 2.1.

Πίνακας 2.1: Σύγκριση μόνιμων και μη μόνιμων εικονικών επιφανειών εργασίας.

Χαρακτηριστικό	Persistent VDI	Non-persistent VDI
Συσχέτιση χρήστη με περιβάλλον	Συνήθως σταθερή αντιστοίχιση του χρήστη με συγκεκριμένο εικονικό περιβάλλον.	Δεν απαιτείται μόνιμη αντιστοίχιση του χρήστη με συγκεκριμένη εικονική μηχανή.
Διατήρηση κατάστασης	Οι αλλαγές μπορούν να διατηρούνται μεταξύ διαδοχικών συνεδριών.	Το περιβάλλον μπορεί να επανέρχεται μετά τη χρήση σε προκαθορισμένη κατάσταση.
Εξατομίκευση	Υποστηρίζει μεγαλύτερο βαθμό εξατομίκευσης ρυθμίσεων, εφαρμογών και δεδομένων.	Η εξατομίκευση είναι περισσότερο περιορισμένη προς όφελος της ομοιομορφίας.
Οργάνωση πόρων	Το περιβάλλον λειτουργεί περισσότερο ως προσωπικός εικονικός σταθμός.	Οι σταθμοί μπορούν να παρέχονται από κοινή ομάδα διαθέσιμων εικονικών μηχανών.
Διαχειριστική έμφαση	Διατήρηση της κατάστασης και των ιδιαίτερων χαρακτηριστικών κάθε περιβάλλοντος.	Επαναχρησιμοποίηση και διατήρηση κοινής, προβλέψιμης κατάστασης.
Ενδεικτικό πεδίο χρήσης	Περιβάλλοντα όπου απαιτείται σταθερό προσωπικό desktop και σημαντική εξατομίκευση.	Κοινόχρηστα περιβάλλοντα όπου απαιτούνται ομοιόμορφοι και επαναχρησιμοποιήσιμοι σταθμοί.

2.3 Κεντριοποιημένη Διαχείριση Συστημάτων

2.3.1 Παραδοσιακή Διαχείριση Σταθμών Εργασίας

Σε ένα περιβάλλον όπου οι σταθμοί εργασίας λειτουργούν ως ανεξάρτητα συστήματα, σημαντικό μέρος της διαχείρισης πραγματοποιείται σε επίπεδο κάθε υπολογιστή. Η εγκατάσταση και ενημέρωση εφαρμογών, η εφαρμογή ρυθμίσεων, η αντιμετώπιση προβλημάτων και η επαναφορά ενός συστήματος σε επιθυμητή κατάσταση πρέπει να εκτελούνται ή να ελέγχονται ξεχωριστά για κάθε σταθμό.

Η προσέγγιση αυτή είναι διαχειρίσιμη όταν ο αριθμός των συστημάτων είναι περιορισμένος. Καθώς όμως το πλήθος τους αυξάνεται, οι ίδιες επαναλαμβανόμενες ενέργειες πρέπει να εφαρμόζονται σε περισσότερους υπολογιστές, αυξάνοντας τον απαιτούμενο χρόνο και την πιθανότητα αποκλίσεων μεταξύ των επιμέρους συστημάτων. Ακόμη και όταν οι σταθμοί έχουν αρχικά παρόμοια διαμόρφωση, διαφορετικές ενημερώσεις, αλλαγές χρηστών ή μεμονωμένες παρεμβάσεις μπορούν σταδιακά να οδηγήσουν σε ανομοιογενείς καταστάσεις.

Το πρόβλημα είναι ιδιαίτερα εμφανές σε κοινόχρηστες υποδομές, στις οποίες πολλοί σταθμοί προορίζονται να παρέχουν αντίστοιχο περιβάλλον εργασίας. Σε αυτή την περίπτωση, η ανεξάρτητη διαχείριση κάθε συστήματος δεν αυξάνει μόνο τον συνολικό φόρτο εργασίας, αλλά δυσχεραίνει και τη διατήρηση μιας κοινής και προβλέψιμης κατάστασης σε ολόκληρη την υποδομή[12], [13].

2.3.2 Κεντριοποιημένη Διαχείριση

Η κεντριοποιημένη διαχείριση επιδιώκει τη μεταφορά μέρους των λειτουργιών ελέγχου και διαμόρφωσης από τους επιμέρους σταθμούς σε κοινά σημεία διαχείρισης. Αντί κάθε σύστημα να αντιμετωπίζεται αποκλειστικά ως ανεξάρτητη μονάδα, οι διαθέσιμοι πόροι μπορούν να οργανώνονται σε ομάδες και να υπόκεινται σε κοινές διαδικασίες και πολιτικές.

Η προσέγγιση αυτή διευκολύνει την εφαρμογή ομοιόμορφων ρυθμίσεων και τη συντήρηση μεγαλύτερου αριθμού συστημάτων. Μία αλλαγή μπορεί να σχεδιάζεται και να εφαρμόζεται σε επίπεδο υποδομής, περιορίζοντας την ανάγκη επανάληψης της ίδιας διαδικασίας χειροκίνητα σε κάθε σταθμό. Παράλληλα, η συγκέντρωση της διαχείρισης παρέχει σαφέστερη εικόνα της κατάστασης των διαθέσιμων πόρων και διευκολύνει την εφαρμογή κοινών κανόνων λειτουργίας[14].

Η κεντριοποίηση δεν προϋποθέτει απαραίτητα τη χρήση εικονικοποίησης. Μπορεί να εφαρμοστεί και σε φυσικά συστήματα μέσω εργαλείων απομακρυσμένης διαχείρισης, υπηρεσιών καταλόγου, πολιτικών διαμόρφωσης ή συστημάτων διανομής λογισμικού. Η εικονικοποίηση, ωστόσο, επεκτείνει τις δυνατότητες της προσέγγισης, καθώς επιτρέπει να μεταφερθεί στην κεντρική υποδομή όχι μόνο ο έλεγχος αλλά και η ίδια η εκτέλεση των διαχειριζόμενων υπολογιστικών περιβαλλόντων.

Στην περίπτωση του VDI, η σχέση αυτή είναι ιδιαίτερα άμεση. Οι επιφάνειες εργασίας μπορούν να δημιουργούνται, να οργανώνονται και να συντηρούνται στην κεντρική υποδομή, ενώ τα τερματικά των χρηστών περιορίζονται κυρίως στην πρόσβαση προς αυτές. Έτσι, η διαχείριση του περιβάλλοντος εργασίας μπορεί να αποσυνδεθεί σε σημαντικό βαθμό από τη διαχείριση του φυσικού τερματικού.

2.3.3 Αυτοματοποίηση Διαχείρισης

Η κεντριοποίηση δημιουργεί τις προϋποθέσεις για αποτελεσματικότερη διαχείριση, δεν εξαλείφει όμως από μόνη της την ανάγκη εκτέλεσης επαναλαμβανόμενων ενεργειών. Εάν οι ίδιες λειτουργίες εξακολουθούν να πραγματοποιούνται χειροκίνητα από έναν κεντρικό διαχειριστή, το σημείο από το οποίο εκτελούνται έχει αλλάξει, αλλά ένα σημαντικό μέρος του διαχειριστικού κόστους παραμένει. Για τον λόγο αυτό, η αυτοματοποίηση αποτελεί φυσική συνέχεια της κεντριοποιημένης διαχείρισης.

Ως αυτοματοποίηση διαχείρισης μπορεί να θεωρηθεί η μετατροπή επαναλαμβανόμενων διοικητικών διαδικασιών σε προκαθορισμένες και επαναλήψιμες λειτουργίες που εκτελούνται από λογισμικό[15]. Τέτοιες λειτουργίες μπορούν να περιλαμβάνουν τη δημιουργία πόρων, την εφαρμογή κοινών ρυθμίσεων, την επιλογή διαθέσιμων συστημάτων, τον έλεγχο της κατάστασής τους και την επαναφορά τους σε προκαθορισμένη κατάσταση.

Η αξία της αυτοματοποίησης δεν περιορίζεται στη μείωση του χρόνου εκτέλεσης μιας εργασίας. Η χρήση προκαθορισμένων διαδικασιών μπορεί επίσης να βελτιώσει την επαναληψιμότητα των ενεργειών και να περιορίσει αποκλίσεις που προκύπτουν από διαφορετικούς τρόπους χειροκίνητης εκτέλεσης της ίδιας διαδικασίας. Παράλληλα, οι αυτοματοποιημένες λειτουργίες μπορούν να βασίζονται σε ελέγχους της τρέχουσας κατάστασης και να εφαρμόζουν προκαθορισμένες ενέργειες ανάλογα με τα αποτελέσματα αυτών των ελέγχων, αντί να εκτελούν απλώς μία στατική ακολουθία εντολών[16].

Η αυτοματοποίηση αποκτά ιδιαίτερη σημασία όταν οι διαχειριζόμενοι πόροι είναι εικονικοί. Ενέργειες όπως η δημιουργία, η εκκίνηση, η διακοπή, η επιλογή, η επαναφορά ή η κατάργηση εικονικών μηχανών μπορούν να εκτελούνται μέσω προγραμματιστικών διεπαφών ή εργαλείων γραμμής εντολών και επομένως να ενσωματώνονται σε αυτοματοποιημένες ροές εργασίας. Με αυτόν τον τρόπο, η εικονικοποίηση δεν παρέχει μόνο ευελιξία στην κατανομή των πόρων, αλλά δημιουργεί και ένα περιβάλλον στο οποίο σημαντικό μέρος του κύκλου ζωής τους μπορεί να ελέγχεται μέσω λογισμικού.

2.3.4 Πλεονεκτήματα και Περιορισμοί της Κεντριοποιημένης Διαχείρισης

Η κεντριοποιημένη και αυτοματοποιημένη διαχείριση μπορεί να περιορίσει σημαντικά τις επαναλαμβανόμενες εργασίες που απαιτούνται για τη συντήρηση μεγάλου αριθμού συστημάτων. Η εφαρμογή κοινών διαδικασιών διευκολύνει τη διατήρηση ομοιομορφίας, ενώ η κεντρική εποπτεία παρέχει καλύτερη εικόνα της κατάστασης των διαθέσιμων πόρων. Σε εικονικοποιημένα περιβάλλοντα, οι δυνατότητες αυτές μπορούν να επεκταθούν σε ολόκληρο τον κύκλο ζωής των εικονικών συστημάτων.

Παράλληλα, η κεντριοποίηση μεταφέρει μεγαλύτερο μέρος της εξάρτησης στην κοινή υποδομή. Προβλήματα σε κεντρικούς εξυπηρετητές, στο δίκτυο ή σε υπηρεσίες που χρησιμοποιούνται από πολλούς σταθμούς μπορούν να επηρεάσουν ταυτόχρονα μεγαλύτερο αριθμό χρηστών. Για τον λόγο αυτό, η κεντρική διαχείριση δεν εξαλείφει την ανάγκη για αξιοπιστία, παρακολούθηση και κατάλληλους μηχανισμούς προστασίας· αντίθετα, καθιστά τις απαιτήσεις αυτές σημαντικότερες σε επίπεδο υποδομής.

Η αυτοματοποίηση εισάγει επίσης δικές της απαιτήσεις. Μία λανθασμένη αυτοματοποιημένη ενέργεια μπορεί να εφαρμοστεί γρήγορα σε πολλούς πόρους, επομένως οι διαδικασίες μαζικής

διαχείρισης πρέπει να περιλαμβάνουν κατάλληλους ελέγχους, σαφή όρια ως προς τους πόρους στους οποίους επιτρέπεται να ενεργήσουν και προβλέψιμη συμπεριφορά σε περίπτωση αποτυχίας. Η αυτοματοποίηση δεν πρέπει συνεπώς να αντιμετωπίζεται μόνο ως μέσο ταχύτερης εκτέλεσης ενεργειών, αλλά ως διαδικασία που απαιτεί ελεγχόμενη και ασφαλή σχεδίαση.

Για περιβάλλοντα όπως τα εκπαιδευτικά εργαστήρια, τα πλεονεκτήματα της κεντρικής διαχείρισης είναι ιδιαίτερα σχετικά λόγω του αριθμού των κοινόχρηστων σταθμών και της ανάγκης διατήρησης επαναλαμβανόμενων και προβλέψιμων περιβαλλόντων εργασίας. Ταυτόχρονα, οι περιορισμοί της κεντροκοποίησης και της αυτοματοποίησης πρέπει να λαμβάνονται υπόψη κατά τη σχεδίαση της αντίστοιχης υποδομής, ώστε η μείωση της χειροκίνητης διαχείρισης να μη δημιουργεί νέες ανεξέλεγκτες εξαρτήσεις ή λειτουργικούς κινδύνους.

2.4 Απομακρυσμένη Πρόσβαση σε Επιφάνεια Εργασίας

2.4.1 Έννοια της Απομακρυσμένης Πρόσβασης

Η απομακρυσμένη πρόσβαση σε επιφάνεια εργασίας επιτρέπει σε έναν χρήστη να αλληλεπιδρά μέσω δικτύου με ένα υπολογιστικό περιβάλλον που εκτελείται σε διαφορετικό σύστημα. Το λειτουργικό σύστημα και οι εφαρμογές εκτελούνται στον απομακρυσμένο υπολογιστή, ενώ η τοπική συσκευή χρησιμοποιείται για την παρουσίαση του περιβάλλοντος και τη μεταφορά της αλληλεπίδρασης του χρήστη[17].

Η επικοινωνία αυτή πραγματοποιείται μέσω πρωτοκόλλων απομακρυσμένης επιφάνειας εργασίας. Σε γενικές γραμμές, το απομακρυσμένο σύστημα μεταφέρει προς τον πελάτη τις πληροφορίες που απαιτούνται για την παρουσίαση της συνεδρίας, ενώ ο πελάτης μεταφέρει προς το απομακρυσμένο σύστημα δεδομένα εισόδου, όπως ενέργειες πληκτρολογίου και ποντικιού. Ανάλογα με το πρωτόκολλο και την υλοποίηση, μπορούν επίσης να υποστηρίζονται πρόσθετες λειτουργίες, όπως μεταφορά ήχου, κοινή χρήση προχείρου και ανακατεύθυνση τοπικών συσκευών.

Η απομακρυσμένη πρόσβαση αποτελεί βασικό στοιχείο μιας αρχιτεκτονικής VDI, καθώς επιτρέπει τον διαχωρισμό μεταξύ του συστήματος στο οποίο εκτελείται η επιφάνεια εργασίας και της συσκευής από την οποία χρησιμοποιείται. Το πρωτόκολλο απομακρυσμένης πρόσβασης δεν εκτελεί το ίδιο την εικονικοποίηση ούτε αναλαμβάνει απαραίτητα την επιλογή του κατάλληλου περιβάλλοντος εργασίας. Ο ρόλος του είναι να παρέχει το κανάλι μέσω του οποίου ο τελικός χρήστης αλληλεπιδρά με το περιβάλλον που του έχει διατεθεί.

Η ποιότητα μιας απομακρυσμένης συνεδρίας εξαρτάται μεταξύ άλλων από τα χαρακτηριστικά του δικτύου, όπως η καθυστέρηση, η διαθέσιμη χωρητικότητα και η αξιοπιστία της σύνδεσης, καθώς και από τον τρόπο με τον οποίο το αντίστοιχο πρωτόκολλο μεταφέρει και συμπιέζει τα δεδομένα της συνεδρίας. Για τον λόγο αυτό, η δικτυακή υποδομή αποτελεί σημαντικό παράγοντα για την πρακτική λειτουργία συστημάτων απομακρυσμένης επιφάνειας εργασίας[17].

2.4.2 Remote Desktop Protocol

Το Remote Desktop Protocol (RDP) είναι πρωτόκολλο απομακρυσμένης πρόσβασης που αναπτύχθηκε από τη Microsoft και χρησιμοποιείται ευρέως για την πρόσβαση σε απομακρυσμένα περιβάλλοντα Windows. Βασίζεται σε αρχιτεκτονική πελάτη--εξυπηρετητή, στην οποία το απο-

μακρυσμένο σύστημα παρέχει τη συνεδρία και ο πελάτης αναλαμβάνει την παρουσίαση και την αλληλεπίδραση του χρήστη [18].

Το RDP δεν περιορίζεται αποκλειστικά στη μεταφορά μιας στατικής εικόνας της απομακρυσμένης οθόνης. Το πρωτόκολλο υποστηρίζει μηχανισμούς για τη μεταφορά γραφικών και δεδομένων εισόδου, καθώς και πρόσθετες δυνατότητες που μπορούν να περιλαμβάνουν ήχο, πρόχειρο, τοπικούς αποθηκευτικούς πόρους και άλλες συσκευές, ανάλογα με τις δυνατότητες και τις ρυθμίσεις του πελάτη και του εξυπηρετητή [18], [19].

Η λειτουργία του RDP βασίζεται συνήθως στη δημιουργία μιας δικτυακής συνεδρίας μεταξύ πελάτη και απομακρυσμένου συστήματος. Η TCP θύρα 3389 χρησιμοποιείται κατά κανόνα από τις υπηρεσίες RDP, αν και η συγκεκριμένη ρύθμιση μπορεί να μεταβληθεί. Η πρόσβαση προϋποθέτει επίσης κατάλληλη αυθεντικοποίηση και πολιτικές ασφαλείας, καθώς η δυνατότητα απομακρυσμένης εισόδου σε ένα σύστημα αποτελεί σημαντικό σημείο ελέγχου της υποδομής[20], [21].

Παρότι το RDP συνδέεται κυρίως με περιβάλλοντα Windows, υπάρχουν υλοποιήσεις πελάτη και εξυπηρετητή και για άλλα λειτουργικά συστήματα. Αυτό επιτρέπει τη χρήση του ίδιου βασικού πρωτοκόλλου σε ετερογενή περιβάλλοντα, χωρίς να απαιτείται το τερματικό πρόσβασης να χρησιμοποιεί το ίδιο λειτουργικό σύστημα με το απομακρυσμένο περιβάλλον. Οι συγκεκριμένες υλοποιήσεις που αξιοποιούνται στην παρούσα εργασία παρουσιάζονται στο επόμενο κεφάλαιο.

2.4.3 Εναλλακτικά Πρωτόκολλα Απομακρυσμένης Πρόσβασης

Το RDP δεν αποτελεί τη μοναδική τεχνολογία για απομακρυσμένη πρόσβαση σε γραφικό περιβάλλον. Διαφορετικά πρωτόκολλα και συστήματα έχουν αναπτυχθεί με διαφορετικές αρχιτεκτονικές, δυνατότητες και πεδία εφαρμογής.

Μία διαδεδομένη προσέγγιση είναι το Virtual Network Computing (VNC), το οποίο βασίζεται στο Remote Framebuffer (RFB) protocol. Σε αντίθεση με προσεγγίσεις που μεταφέρουν πληροφορίες υψηλότερου επιπέδου σχετικά με τη συνεδρία, το RFB επικεντρώνεται στη μετάδοση της κατάστασης ενός απομακρυσμένου framebuffer και στη μεταφορά των ενεργειών εισόδου του χρήστη [22]. Η σχετική απλότητα και ανεξαρτησία από συγκεκριμένο λειτουργικό σύστημα έχουν οδηγήσει σε πολλές διαφορετικές υλοποιήσεις VNC.

Μία ακόμη τεχνολογία που συναντάται σε περιβάλλοντα εικονικοποίησης είναι το SPICE (Simple Protocol for Independent Computing Environments). Το SPICE σχεδιάστηκε για απομακρυσμένη αλληλεπίδραση με εικονικές μηχανές και υποστηρίζει λειτουργίες που σχετίζονται με γραφικά, είσοδο χρήστη, ήχο και ανακατεύθυνση συσκευών [23]. Η στενή του σχέση με εικονικοποιημένα περιβάλλοντα το διαφοροποιεί από γενικότερες λύσεις απομακρυσμένου ελέγχου.

Η επιλογή πρωτοκόλλου εξαρτάται από τις απαιτήσεις κάθε υποδομής, την υποστήριξη των λειτουργικών συστημάτων, τις απαιτήσεις απόδοσης και ασφάλειας και τις δυνατότητες των διαθέσιμων πελατών και εξυπηρετητών. Για την παρούσα εργασία επιλέχθηκε η οικογένεια τεχνολογιών που βασίζεται στο RDP, ενώ οι συγκεκριμένες υλοποιήσεις και ο τρόπος ένταξής τους στο σύστημα εξετάζονται στο Κεφάλαιο 3.

2.5 Thin Clients

2.5.1 Έννοια και Χαρακτηριστικά

Ο όρος thin client αναφέρεται σε έναν σταθμό πρόσβασης του οποίου σημαντικό μέρος της υπολογιστικής λειτουργίας παρέχεται από απομακρυσμένη υποδομή και όχι από το ίδιο το τερματικό. Σε αντίθεση με έναν συμβατικό προσωπικό υπολογιστή, στον οποίο το λειτουργικό περιβάλλον και οι εφαρμογές εκτελούνται κυρίως τοπικά, ένας thin client χρησιμοποιείται κατά κύριο λόγο για την πρόσβαση σε υπηρεσίες ή περιβάλλοντα που εκτελούνται σε κεντρικά συστήματα [24].

Η ακριβής λειτουργικότητα ενός thin client μπορεί να διαφέρει ανάλογα με την αρχιτεκτονική στην οποία εντάσσεται. Σε ένα περιβάλλον απομακρυσμένης επιφάνειας εργασίας, το τερματικό πρέπει τουλάχιστον να διαθέτει το απαραίτητο τοπικό λειτουργικό περιβάλλον, δικτυακή συνδεσιμότητα και λογισμικό πελάτη για το χρησιμοποιούμενο πρωτόκολλο. Η κύρια εκτέλεση των εφαρμογών και η επεξεργασία του περιβάλλοντος εργασίας πραγματοποιούνται όμως στον απομακρυσμένο υπολογιστικό πόρο.

Η προσέγγιση αυτή επιτρέπει τη μείωση της εξάρτησης του περιβάλλοντος εργασίας από τα χαρακτηριστικά του τερματικού. Τοπικά απαιτούνται κυρίως οι πόροι που είναι αναγκαίοι για τη λειτουργία του συστήματος πρόσβασης και την παρουσίαση της απομακρυσμένης συνεδρίας, ενώ οι απαιτήσεις των εφαρμογών εξυπηρετούνται από την κεντρική υποδομή.

Οι thin clients συνδέονται επομένως στενά με αρχιτεκτονικές VDI, χωρίς όμως οι δύο έννοιες να ταυτίζονται. Το VDI περιγράφει τον τρόπο με τον οποίο παρέχονται και διαχειρίζονται οι εικονικές επιφάνειες εργασίας, ενώ ο thin client αποτελεί μία πιθανή κατηγορία τερματικού μέσω του οποίου ο χρήστης αποκτά πρόσβαση σε αυτές.

2.5.2 Πλεονεκτήματα και Περιορισμοί

Ένα βασικό πλεονέκτημα των thin clients είναι ότι μειώνουν τις λειτουργικές απαιτήσεις που τοποθετούνται στον τελικό σταθμό. Εφόσον οι κύριες εφαρμογές εκτελούνται στην απομακρυσμένη υποδομή, δεν είναι απαραίτητο κάθε τερματικό να διαθέτει την υπολογιστική ισχύ ή τη διαμόρφωση που θα απαιτούνταν για την τοπική εκτέλεση του πλήρους περιβάλλοντος εργασίας. Αυτό μπορεί να επιτρέψει τη χρήση απλούστερου εξοπλισμού και, σε ορισμένες περιπτώσεις, την αξιοποίηση συστημάτων που δεν θα ήταν πλέον επαρκή για την τοπική εκτέλεση σύγχρονων εφαρμογών.

Παράλληλα, ο περιορισμένος ρόλος του τερματικού μπορεί να απλοποιήσει τη συντήρησή του. Οι αλλαγές στο κεντρικά παρεχόμενο περιβάλλον εργασίας δεν απαιτείται να αναπαράγονται ως πλήρεις τοπικές εγκαταστάσεις σε κάθε thin client. Ωστόσο, το πλεονέκτημα αυτό δεν εξαιλεί πλήρως τη διαχείριση των τερματικών, καθώς εξακολουθούν να απαιτούν συντήρηση του τοπικού λειτουργικού συστήματος, του λογισμικού απομακρυσμένης πρόσβασης και της δικτυακής τους διαμόρφωσης.

Η μεγαλύτερη εξάρτηση από την κεντρική υποδομή και το δίκτυο αποτελεί ταυτόχρονα βασικό περιορισμό. Μία αστοχία στη δικτυακή σύνδεση μπορεί να καταστήσει το απομακρυσμένο περιβάλλον μη προσβάσιμο, ενώ προβλήματα στην κεντρική υποδομή μπορούν να επηρεάσουν ταυτόχρονα πολλούς σταθμούς. Επιπλέον, η εμπειρία του χρήστη εξαρτάται από την απόδοση τόσο της κεντρικής υποδομής όσο και της δικτυακής επικοινωνίας.

Για τον λόγο αυτό, η χρήση thin clients δεν πρέπει να θεωρείται ότι μεταφέρει χωρίς κόστος όλες τις απαιτήσεις ενός σταθμού εργασίας σε έναν εξυπηρετητή. Αντίθετα, μεταβάλλει την κατανομή των απαιτήσεων: μειώνει τη σημασία της τοπικής υπολογιστικής ισχύος για την εκτέλεση εφαρμογών, αλλά αυξάνει τη σημασία της κεντρικής χωρητικότητας, της δικτυακής συνδεσιμότητας και της διαθεσιμότητας των κοινών υπηρεσιών[17], [24].

2.5.3 Thin Clients σε Εκπαιδευτικά Εργαστήρια

Η χρήση thin clients παρουσιάζει ιδιαίτερο ενδιαφέρον σε εκπαιδευτικά εργαστήρια, όπου ένας μεγάλος αριθμός σταθμών χρησιμοποιείται για πρόσβαση σε κοινά ή προκαθορισμένα περιβάλλοντα εργασίας. Ο περιορισμός των λειτουργιών που πρέπει να εκτελούνται τοπικά μπορεί να διευκολύνει τη διατήρηση απλούστερης διαμόρφωσης στους φυσικούς σταθμούς, ενώ οι απαιτητικότερες εφαρμογές και τα λειτουργικά περιβάλλοντα μπορούν να παρέχονται από κεντρικούς υπολογιστικούς πόρους[25].

Μία πιθανή συνέπεια της προσέγγισης είναι η δυνατότητα παράτασης της αξιοποίησης παλαιότερου εξοπλισμού. Ένας υπολογιστής που δεν διαθέτει πλέον επαρκείς πόρους για την τοπική εκτέλεση του απαιτούμενου εκπαιδευτικού λογισμικού ενδέχεται να εξακολουθεί να μπορεί να λειτουργήσει ως τερματικό απομακρυσμένης πρόσβασης. Η δυνατότητα αυτή εξαρτάται από τις πραγματικές απαιτήσεις του λογισμικού πελάτη, τα χαρακτηριστικά του υλικού και του δικτύου και τις απαιτήσεις της απομακρυσμένης συνεδρίας και επομένως δεν μπορεί να θεωρείται δεδομένη για κάθε παλιό σύστημα[25].

Στο πλαίσιο της παρούσας εργασίας, ο thin client εξετάζεται ως το σημείο πρόσβασης του χρήστη προς την κεντρικά παρεχόμενη επιφάνεια εργασίας. Η συγκεκριμένη υλοποίηση και η αλληλεπίδρασή της με τα υπόλοιπα στοιχεία του συστήματος παρουσιάζονται στα επόμενα κεφάλαια.

2.6 Σύνοψη Κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της προσέγγισης που ακολουθείται στην παρούσα εργασία. Η εικονικοποίηση επιτρέπει την εκτέλεση πολλαπλών απομονωμένων υπολογιστικών περιβαλλόντων πάνω σε κοινή φυσική υποδομή, ενώ η εικονικοποίηση επιφάνειας εργασίας επεκτείνει την αρχή αυτή στον τρόπο με τον οποίο παρέχονται τα περιβάλλοντα των τελικών χρηστών. Στο πλαίσιο του VDI, οι επιφάνειες εργασίας μπορούν να εκτελούνται κεντρικά και να διατίθενται απομακρυσμένα, με δυνατότητα οργάνωσης τόσο μόνιμων όσο και μη μόνιμων περιβαλλόντων ανάλογα με τις απαιτήσεις της υποδομής.

Η συγκέντρωση των υπολογιστικών περιβαλλόντων δημιουργεί παράλληλα τις προϋποθέσεις για κεντρικοποιημένη και αυτοματοποιημένη διαχείριση. Η κεντρικοποίηση επιτρέπει την εφαρμογή κοινών διαδικασιών σε πολλαπλούς πόρους, ενώ η αυτοματοποίηση μπορεί να μετατρέψει επαναλαμβανόμενες διοικητικές ενέργειες σε ελεγχόμενες και επαναλήψιμες λειτουργίες. Η απομακρυσμένη πρόσβαση και οι thin clients συμπληρώνουν το μοντέλο αυτό, επιτρέποντας τον διαχωρισμό μεταξύ του συστήματος στο οποίο εκτελείται το περιβάλλον εργασίας και του τερματικού μέσω του οποίου χρησιμοποιείται.

Οι έννοιες αυτές διαμορφώνουν τη θεωρητική βάση για την ανάπτυξη ενός συστήματος στο

οποίο πολλαπλά περιβάλλοντα εργασίας μπορούν να παρέχονται και να διαχειρίζονται από κοινή υποδομή, αντί να συντηρούνται αποκλειστικά ως ανεξάρτητοι φυσικοί σταθμοί. Στο επόμενο κεφάλαιο παρουσιάζονται οι συγκεκριμένες τεχνολογίες που επιλέχθηκαν για την υλοποίηση της προσέγγισης αυτής και ο ρόλος τους στο συνολικό σύστημα.

Κεφάλαιο 3

Τεχνολογίες και Εργαλεία

3.1 Εισαγωγή

Η ανάπτυξη ενός συστήματος εικονικών επιφανειών εργασίας προϋποθέτει τη συνεργασία διαφορετικών τεχνολογιών για την εικονικοποίηση των υπολογιστικών πόρων, την απομακρυσμένη πρόσβαση, την επικοινωνία μεταξύ των επιμέρους συστημάτων και την αυτοματοποίηση των διαδικασιών διαχείρισης. Η επιλογή των κατάλληλων εργαλείων επηρεάζει τόσο τις δυνατότητες της τελικής υποδομής όσο και τον τρόπο με τον οποίο μπορούν να αυτοματοποιηθούν οι λειτουργίες της.

Κεντρικό ρόλο στην παρούσα εργασία κατέχει το Proxmox Virtual Environment (Proxmox VE), το οποίο παρέχει την υποδομή δημιουργίας και διαχείρισης των εικονικών μηχανών μέσω QEMU/KVM. Συμπληρωματικά, το QEMU Guest Agent παρέχει μηχανισμό επικοινωνίας μεταξύ της πλατφόρμας εικονικοποίησης και των φιλοξενούμενων λειτουργικών συστημάτων. Για την απομακρυσμένη πρόσβαση χρησιμοποιείται το Remote Desktop Protocol (RDP), με το FreeRDP στην πλευρά του τερματικού και το XRDP για την παροχή αντίστοιχης υπηρεσίας στα συστήματα GNU/Linux.

Οι λειτουργίες αυτοματοποίησης βασίζονται κυρίως σε εργαλεία του περιβάλλοντος GNU/Linux, στο OpenSSH και σε σενάρια κελύφους Bash. Οι τεχνολογίες αυτές επιτρέπουν την προγραμματιστική εκτέλεση λειτουργιών διαχείρισης, την επικοινωνία μεταξύ του thin client και της κεντρικής υποδομής και τον συντονισμό επαναλαμβανόμενων ενεργειών χωρίς να απαιτείται κάθε φορά χειροκίνητη παρέμβαση του διαχειριστή.

Στο παρόν κεφάλαιο παρουσιάζονται τα βασικά χαρακτηριστικά των τεχνολογιών αυτών και ο ρόλος που τις καθιστά κατάλληλες για την ανάπτυξη του συστήματος. Η αρχιτεκτονική με την οποία συνδυάζονται παρουσιάζεται στο Κεφάλαιο 4, ενώ οι συγκεκριμένες ρυθμίσεις και οι μηχανισμοί μέσω των οποίων αξιοποιούνται στην πειραματική υλοποίηση περιγράφονται στο Κεφάλαιο 5.

3.2 Proxmox Virtual Environment

3.2.1 Γενική Παρουσίαση

Το Proxmox Virtual Environment (Proxmox VE) αποτελεί πλατφόρμα ανοικτού κώδικα για την εικονικοποίηση και την κεντρική διαχείριση υπολογιστικών υποδομών. Βασίζεται στο Debian GNU/Linux και ενσωματώνει σε κοινό περιβάλλον λειτουργίες για τη δημιουργία και διαχείριση εικονικών μηχανών, containers, αποθηκευτικών πόρων και εικονικών δικτύων. Παρέχει επίσης λειτουργίες όπως δημιουργία στιγμιότυπων και αντιγράφων ασφαλείας, κλωνοποίηση εικονικών μηχανών και υποστήριξη εγκαταστάσεων πολλαπλών κόμβων [26], [27].

Για την εικονικοποίηση υπολογιστικών συστημάτων, το Proxmox VE αξιοποιεί τις τεχνολογίες QEMU/KVM, μέσω των οποίων μπορούν να εκτελούνται ανεξάρτητες εικονικές μηχανές με διαφορετικά φιλοξενούμενα λειτουργικά συστήματα. Παράλληλα υποστηρίζει Linux Containers (LXC), τα οποία χρησιμοποιούν εικονικοποίηση σε επίπεδο λειτουργικού συστήματος [26]. Για την παρούσα εργασία ιδιαίτερο ενδιαφέρον παρουσιάζουν οι εικονικές μηχανές QEMU/KVM, καθώς αυτές χρησιμοποιούνται για την παροχή των εικονικών σταθμών εργασίας.

Η διαχείριση της πλατφόρμας μπορεί να πραγματοποιηθεί μέσω του ενσωματωμένου γραφικού περιβάλλοντος Ιστού, εργαλείων γραμμής εντολών και REST API [27]. Η ύπαρξη προγραμματιστικών διεπαφών είναι ιδιαίτερα σημαντική για αυτοματοποιημένα συστήματα, καθώς επιτρέπει την εκτέλεση λειτουργιών διαχείρισης χωρίς να απαιτείται αλληλεπίδραση με το γραφικό περιβάλλον.

Το Proxmox VE διατίθεται ως λογισμικό ανοικτού κώδικα υπό την άδεια GNU Affero General Public License έκδοση 3 (GNU AGPLv3) [26]. Ο συνδυασμός των δυνατοτήτων εικονικοποίησης, των μηχανισμών κεντρικής διαχείρισης και των διαθέσιμων διεπαφών αυτοματοποίησης το καθιστά κατάλληλη τεχνολογική βάση για την πειραματική υποδομή που εξετάζεται στην παρούσα εργασία.

3.2.2 Αρχιτεκτονική και Βασικά Χαρακτηριστικά

Ένας εξυπηρετητής στον οποίο εκτελείται το Proxmox VE αποτελεί έναν κόμβο (node) της πλατφόρμας και παρέχει τους υπολογιστικούς, δικτυακούς και αποθηκευτικούς πόρους που χρησιμοποιούνται από τις εικονικές μηχανές. Ένας κόμβος μπορεί να λειτουργεί αυτόνομα ή να συμμετέχει σε συστοιχία πολλαπλών κόμβων, ενώ η διαχείριση των εικονικών πόρων πραγματοποιείται μέσω των υπηρεσιών του Proxmox VE [27].

Για τη δικτύωση των εικονικών συστημάτων, το Proxmox VE αξιοποιεί τους μηχανισμούς δικτύωσης του Linux και υποστηρίζει εικονικές γέφυρες (Linux bridges). Μία τέτοια γέφυρα μπορεί να συνδέει τις εικονικές δικτυακές διεπαφές των φιλοξενούμενων συστημάτων με φυσικές διεπαφές του host ή με άλλα εικονικά δίκτυα. Με τον τρόπο αυτό, οι εικονικές μηχανές μπορούν να αποκτούν πρόσβαση στο φυσικό δίκτυο ή να τοποθετούνται σε ξεχωριστά εικονικά τμήματα, ανάλογα με τη διαμόρφωση της υποδομής [28].

Η πλατφόρμα υποστηρίζει επίσης διαφορετικούς μηχανισμούς αποθήκευσης για τους δίσκους και τα υπόλοιπα δεδομένα των εικονικών μηχανών. Ανάλογα με τις απαιτήσεις της εγκατάστασης μπορούν να χρησιμοποιηθούν τοπικοί ή δικτυακοί αποθηκευτικοί πόροι και τεχνολογίες όπως LVM, ZFS, NFS, iSCSI και Ceph [29]. Η επιλογή του μηχανισμού αποθήκευσης επηρε-

άξει και τη διαθεσιμότητα ορισμένων λειτουργιών της πλατφόρμας, όπως τα snapshots και οι συνδεδεμένοι κλώνοι.

Ιδιαίτερη σημασία για τη διαχείριση επαναχρησιμοποιούμενων εικονικών σταθμών έχει η δυνατότητα δημιουργίας στιγμιότυπων (snapshots). Ένα snapshot αποτυπώνει την κατάσταση μιας εικονικής μηχανής σε συγκεκριμένο χρονικό σημείο και μπορεί, όταν υποστηρίζεται από τη χρησιμοποιούμενη διαμόρφωση, να αποτελέσει σημείο επαναφοράς της [27]. Η δυνατότητα αυτή είναι χρήσιμη σε κοινόχρηστα περιβάλλοντα στα οποία απαιτείται ένας εικονικός σταθμός να μπορεί να επιστρέφει σε γνωστή κατάσταση μετά τη χρήση του.

Το Proxmox VE υποστηρίζει επιπλέον τη συγκρότηση συστοιχιών πολλαπλών κόμβων και λειτουργίες που σχετίζονται με μεγαλύτερες εγκαταστάσεις, όπως κεντρική διαχείριση των κόμβων, μετεγκατάσταση εικονικών μηχανών και υψηλή διαθεσιμότητα [30]. Οι δυνατότητες αυτές αποτελούν χαρακτηριστικά της πλατφόρμας, αλλά δεν είναι απαραίτητες για τη λειτουργία ενός μεμονωμένου κόμβου και δεν αποτελούν αντικείμενο της πειραματικής υλοποίησης της παρούσας εργασίας.

3.2.3 QEMU/KVM και Εικονικές Μηχανές

Η εκτέλεση των εικονικών μηχανών στο Proxmox VE βασίζεται στον συνδυασμό των τεχνολογιών Kernel-based Virtual Machine (KVM) και QEMU. Το KVM παρέχει τους βασικούς μηχανισμούς εικονικοποίησης μέσω του πυρήνα Linux και αξιοποιεί τις δυνατότητες εικονικοποίησης του επεξεργαστή, ενώ το QEMU παρέχει το περιβάλλον της εικονικής μηχανής και τις απαραίτητες εικονικές συσκευές [31], [32].

Μέσω του QEMU μπορούν να παρουσιαστούν στο φιλοξενούμενο λειτουργικό σύστημα εικονικοί επεξεργαστές, μνήμη, ελεγκτές αποθήκευσης, δικτυακές διεπαφές και άλλες συσκευές. Σε συνδυασμό με το KVM, η εκτέλεση του φιλοξενούμενου συστήματος μπορεί να αξιοποιεί την υποστήριξη εικονικοποίησης του φυσικού επεξεργαστή, ενώ το QEMU αναλαμβάνει τη διαχείριση του εικονικού υλικού [31], [33].

Για τις λειτουργίες εισόδου και εξόδου μπορούν να χρησιμοποιηθούν παραεικονικοποιημένες συσκευές VirtIO, οι οποίες έχουν σχεδιαστεί ειδικά για εικονικοποιημένα περιβάλλοντα. Η χρήση τους απαιτεί την ύπαρξη κατάλληλων οδηγών στο φιλοξενούμενο λειτουργικό σύστημα και επιτρέπει την αποφυγή πλήρους εξομοίωσης συμβατικών συσκευών για τις αντίστοιχες λειτουργίες [32].

Το Proxmox VE προσθέτει πάνω από το QEMU/KVM το επίπεδο διαχείρισης των εικονικών μηχανών. Κάθε VM αναγνωρίζεται από μοναδικό αριθμητικό αναγνωριστικό (VMID) και η διαμόρφωσή της διαχειρίζεται από την πλατφόρμα. Έτσι, ο διαχειριστής ή ένα αυτοματοποιημένο εργαλείο μπορεί να χειρίζεται τις εικονικές μηχανές μέσω των διεπαφών του Proxmox VE χωρίς να απαιτείται άμεση διαχείριση των αντίστοιχων διεργασιών QEMU [32].

3.2.4 Διαχείριση Εικονικών Μηχανών μέσω Proxmox VE

Το Proxmox VE παρέχει λειτουργίες που καλύπτουν βασικά στάδια του κύκλου ζωής μιας εικονικής μηχανής, όπως δημιουργία, διαμόρφωση, εκκίνηση, τερματισμός, κλωνοποίηση και επαναφορά. Οι λειτουργίες αυτές είναι διαθέσιμες τόσο μέσω του γραφικού περιβάλλοντος όσο και μέσω εργαλείων γραμμής εντολών και του API της πλατφόρμας [34], [35].

Για τις εικονικές μηχανές QEMU/KVM, σημαντικό εργαλείο γραμμής εντολών αποτελεί το

qm. Μέσω αυτού μπορούν να ελεγχθούν η κατάσταση και η διαμόρφωση μιας VM και να εκτελεστούν ενέργειες όπως εκκίνηση, τερματισμός, δημιουργία κλώνων και διαχείριση snapshots [34]. Η δυνατότητα εκτέλεσης αυτών των ενεργειών μέσω γραμμής εντολών είναι ιδιαίτερα σημαντική για την αυτοματοποίηση, καθώς επιτρέπει την ενσωμάτωσή τους σε scripts και άλλες προγραμματισμένες διαδικασίες.

Για τη δημιουργία πολλαπλών εικονικών μηχανών με κοινή αρχική διαμόρφωση, η πλατφόρμα υποστηρίζει πρότυπα (templates) και κλωνοποίηση. Ανάλογα με την υποδομή αποθήκευσης, μπορούν να δημιουργηθούν πλήρεις ή συνδεδεμένοι κλώνοι (full και linked clones), παρέχοντας διαφορετικές επιλογές ως προς την ανεξαρτησία των εικονικών δίσκων και την αξιοποίηση του αποθηκευτικού χώρου [27], [32]. Οι μηχανισμοί αυτοί εξετάζονται αναλυτικότερα στην Ενότητα 3.4.3.

Σε συνδυασμό με τα snapshots, οι παραπάνω λειτουργίες παρέχουν τους βασικούς μηχανισμούς για τη δημιουργία και επαναχρησιμοποίηση τυποποιημένων εικονικών σταθμών εργασίας. Παράλληλα, επειδή οι λειτουργίες διαχείρισης είναι διαθέσιμες μέσω προγραμματιστικών διεπαφών, μπορούν να αποτελέσουν τη βάση για αυτοματοποιημένες διαδικασίες που ενεργούν πάνω σε ομάδες εικονικών μηχανών. Ο συγκεκριμένος τρόπος με τον οποίο οι δυνατότητες αυτές οργανώνονται και αξιοποιούνται στο σύστημα της παρούσας εργασίας παρουσιάζεται στα κεφάλαια σχεδίασης και υλοποίησης.

3.2.5 Λόγοι Επιλογής του Proxmox VE

Για τις ανάγκες της παρούσας εργασίας επιλέχθηκε το Proxmox VE ως βάση της εικονικοποιημένης υποδομής, επειδή συνδυάζει τις απαιτούμενες δυνατότητες εκτέλεσης και διαχείρισης εικονικών μηχανών με διεπαφές κατάλληλες για την ανάπτυξη αυτοματισμών, χωρίς να απαιτείται η χρήση μιας ολοκληρωμένης εμπορικής πλατφόρμας VDI.

Βασικό κριτήριο αποτέλεσε η υποστήριξη εικονικών μηχανών QEMU/KVM με διαφορετικά φιλοξενούμενα λειτουργικά συστήματα, επιτρέποντας την ανάπτυξη περιβαλλόντων τόσο Microsoft Windows όσο και GNU/Linux [26], [32]. Παράλληλα, οι δυνατότητες δημιουργίας προτύπων, κλωνοποίησης και snapshots παρέχουν μηχανισμούς κατάλληλους για την προετοιμασία πολλαπλών σταθμών από κοινή βάση και την επαναφορά τους σε προκαθορισμένη κατάσταση [27], [34].

Εξίσου σημαντική ήταν η δυνατότητα προγραμματιστικής διαχείρισης. Το εργαλείο qm και το REST API επιτρέπουν την ενσωμάτωση λειτουργιών της πλατφόρμας σε εξωτερικούς μηχανισμούς αυτοματοποίησης [34], [35]. Η ιδιότητα αυτή συνδέεται άμεσα με το αντικείμενο της εργασίας, καθώς η διαχείριση πολλαπλών εικονικών σταθμών πρέπει να μπορεί να πραγματοποιείται μέσω επαναλήψιμων διαδικασιών και όχι αποκλειστικά μέσω χειροκίνητων ενεργειών στο γραφικό περιβάλλον.

Τέλος, ο ανοικτός χαρακτήρας του Proxmox VE και η δυνατότητα αξιοποίησης της βασικής λειτουργικότητάς του χωρίς την αγορά άδειας για μια εξειδικευμένη εμπορική λύση VDI ήταν συμβατά με τον πειραματικό και εκπαιδευτικό χαρακτήρα της εργασίας [26]. Η επιλογή της πλατφόρμας δεν υποδηλώνει ότι αποτελεί τη μοναδική ή γενικά βέλτιστη λύση εικονικοποίησης, αλλά ότι παρέχει έναν κατάλληλο συνδυασμό λειτουργιών για τις απαιτήσεις και το πεδίο της συγκεκριμένης υλοποίησης.

3.3 QEMU Guest Agent

3.3.1 Ρόλος του Guest Agent

Το QEMU Guest Agent (QGA) είναι μία υπηρεσία που εκτελείται στο εσωτερικό μιας εικονικής μηχανής και παρέχει έναν μηχανισμό επικοινωνίας μεταξύ του φιλοξενούμενου λειτουργικού συστήματος και του QEMU. Μέσω του μηχανισμού αυτού, το σύστημα φιλοξενίας μπορεί να αποκτά πληροφορίες για την εσωτερική κατάσταση του guest και να ζητά την εκτέλεση συγκεκριμένων υποστηριζόμενων λειτουργιών **proxmox_qemu_agent**, [36].

Η χρήση του Guest Agent επιτρέπει στην πλατφόρμα εικονικοποίησης να αποκτά πληροφορίες οι οποίες δεν είναι πάντοτε διαθέσιμες μόνο από την εξωτερική κατάσταση της εικονικής μηχανής. Για παράδειγμα, το γεγονός ότι μία VM βρίσκεται σε κατάσταση εκτέλεσης δεν σημαίνει απαραίτητα ότι το λειτουργικό της σύστημα έχει ολοκληρώσει την εκκίνηση ή ότι οι δικτυακές της διεπαφές έχουν αποκτήσει τις απαιτούμενες διευθύνσεις. Η ύπαρξη ενός agent στο εσωτερικό του guest παρέχει επομένως ένα πρόσθετο κανάλι παρατήρησης και διαχείρισης.

Το QGA υποστηρίζεται από το Proxmox VE και μπορεί να ενεργοποιηθεί για τις αντίστοιχες εικονικές μηχανές, με την προϋπόθεση ότι η υπηρεσία έχει εγκατασταθεί και εκτελείται στο φιλοξενούμενο λειτουργικό σύστημα [32]. Η δυνατότητα αυτή είναι διαθέσιμη τόσο για συστήματα GNU/Linux όσο και για Microsoft Windows, γεγονός που επιτρέπει τη χρήση κοινού μηχανισμού επικοινωνίας σε ετερογενή σύνολα εικονικών μηχανών.

3.3.2 Επικοινωνία Host--Guest

Η επικοινωνία του QEMU Guest Agent πραγματοποιείται μέσω ειδικού καναλιού μεταξύ του host και του guest. Στην πλευρά της εικονικής μηχανής, η υπηρεσία Guest Agent λαμβάνει αιτήματα μέσω του αντίστοιχου εικονικού καναλιού και αλληλεπιδρά με το φιλοξενούμενο λειτουργικό σύστημα για την εκτέλεση των υποστηριζόμενων λειτουργιών [36].

Οι διαθέσιμες λειτουργίες περιλαμβάνουν, μεταξύ άλλων, την ανάκτηση πληροφοριών για το λειτουργικό σύστημα και τις δικτυακές διεπαφές, καθώς και λειτουργίες που σχετίζονται με τη διαχείριση της κατάστασης του guest. Το QGA παρέχει επίσης δυνατότητα εκτέλεσης εντολών στο εσωτερικό του φιλοξενούμενου συστήματος μέσω των αντίστοιχων μηχανισμών του πρωτοκόλλου του [36].

Η επικοινωνία αυτή είναι διαφορετική από μία συμβατική δικτυακή σύνδεση, όπως SSH ή RDP. Δεν προορίζεται να παρέχει διαδραστική συνεδρία τελικού χρήστη, αλλά να λειτουργεί ως κανάλι μεταξύ της υποδομής εικονικοποίησης και του φιλοξενούμενου λειτουργικού συστήματος. Για τον λόγο αυτό μπορεί να χρησιμοποιηθεί για λειτουργίες διαχείρισης ακόμη και όταν η τελική υπηρεσία απομακρυσμένης πρόσβασης δεν έχει ακόμη καταστεί διαθέσιμη.

Η διάκριση αυτή είναι ιδιαίτερα χρήσιμη σε αυτοματοποιημένες υποδομές. Η εξωτερική κατάσταση μιας εικονικής μηχανής μπορεί να συνδυάζεται με πληροφορίες που προέρχονται από το εσωτερικό του guest, επιτρέποντας στις διαδικασίες διαχείρισης να βασίζονται σε περισσότερα στοιχεία από την απλή κατάσταση εκκίνησης ή τερματισμού της VM.

3.3.3 Αξιοποίηση στο Σύστημα VDI

Σε ένα σύστημα VDI, η εκκίνηση μιας εικονικής μηχανής δεν αρκεί από μόνη της για να θεωρηθεί ότι ο αντίστοιχος σταθμός είναι προσβάσιμος από τον τελικό χρήστη. Μετά την εκκίνηση πρέπει να ολοκληρωθούν οι απαραίτητες λειτουργίες του φιλοξενούμενου λειτουργικού συστήματος και να είναι διαθέσιμη η δικτυακή πληροφορία που απαιτείται για την πραγματοποίηση της απομακρυσμένης σύνδεσης.

Για τον λόγο αυτό, το QEMU Guest Agent επιλέχθηκε ως ο μηχανισμός μέσω του οποίου η κεντρική υποδομή μπορεί να αποκτά πληροφορίες από τις εικονικές μηχανές χωρίς να εξαρτάται από μία ξεχωριστή υπηρεσία απομακρυσμένης διαχείρισης σε κάθε λειτουργικό σύστημα. Ιδιαίτερα σημαντική είναι η δυνατότητα ανάκτησης πληροφοριών για τις δικτυακές διεπαφές του guest, καθώς επιτρέπει σε έναν αυτοματοποιημένο μηχανισμό να προσδιορίζει τη διεύθυνση μέσω της οποίας μπορεί στη συνέχεια να πραγματοποιηθεί η σύνδεση του χρήστη.

Η χρήση κοινού Guest Agent σε εικονικές μηχανές Windows και Linux παρέχει επίσης μία ενιαία διεπαφή μεταξύ της πλατφόρμας εικονικοποίησης και διαφορετικών φιλοξενούμενων λειτουργικών συστημάτων. Οι διαφορές μεταξύ των δύο λειτουργικών συστημάτων εξακολουθούν να υφίστανται σε επίπεδο προετοιμασίας και παροχής της επιφάνειας εργασίας, αλλά η κεντρική υποδομή μπορεί να χρησιμοποιεί τον ίδιο βασικό μηχανισμό για την ανάκτηση πληροφοριών σχετικά με την κατάσταση του guest.

Στην υλοποίηση της παρούσας εργασίας, το QEMU Guest Agent αποτελεί επομένως το διαχειριστικό κανάλι που συνδέει την κατάσταση της εικονικής μηχανής όπως αυτή είναι γνωστή στο Proxmox VE με πληροφορίες που προέρχονται από το εσωτερικό του λειτουργικού συστήματος. Η ακριβής διαδικασία μέσω της οποίας αξιοποιείται για την αναζήτηση της δικτυακής διεύθυνσης και για άλλες λειτουργίες διαχείρισης παρουσιάζεται στο Κεφάλαιο 5.

3.4 Λειτουργικά Συστήματα Εικονικών Σταθμών Εργασίας

3.4.1 Microsoft Windows

Το Microsoft Windows αποτελεί ένα από τα λειτουργικά συστήματα που υποστηρίζονται από τους εικονικούς σταθμούς της παρούσας εργασίας. Η υποστήριξή του είναι σημαντική σε ένα εκπαιδευτικό περιβάλλον, καθώς ένα μέρος του λογισμικού που μπορεί να απαιτείται από μαθήματα ή εργαστηριακές δραστηριότητες διατίθεται για Windows ή χρησιμοποιείται ευρέως σε αυτό το λειτουργικό σύστημα.

Το Windows μπορεί να εκτελεστεί ως φιλοξενούμενο λειτουργικό σύστημα σε εικονικές μηχανές QEMU/KVM μέσω του Proxmox VE. Για την αποτελεσματικότερη λειτουργία του σε εικονικοποιημένο περιβάλλον μπορούν να χρησιμοποιηθούν οι κατάλληλοι οδηγοί VirtIO, ενώ η εγκατάσταση του QEMU Guest Agent επιτρέπει την επικοινωνία του guest με την υποδομή εικονικοποίησης [36], [37].

Ιδιαίτερο ενδιαφέρον για τη δημιουργία επαναχρησιμοποιήσιμων εικονικών σταθμών παρουσιάζει το εργαλείο System Preparation (Sysprep) των Windows. Το Sysprep παρέχει μηχανισμούς προετοιμασίας μιας εγκατάστασης Windows πριν από την αξιοποίησή της ως βάση για νέες εγκαταστάσεις ή εικονικές μηχανές. Με τη διαδικασία γενίκευσης μπορούν να αφαιρεθούν πληροφορίες που είναι μοναδικές για τη συγκεκριμένη εγκατάσταση, ώστε η προετοιμασμένη εικόνα να μπορεί να χρησιμοποιηθεί ως πρότυπο [38].

Η διαδικασία μπορεί να συνδυαστεί με αρχεία απαντήσεων (answer files), μέσω των οποίων καθορίζονται παράμετροι της εγκατάστασης και της αρχικής διαμόρφωσης των Windows [39]. Οι δυνατότητες αυτές είναι κατάλληλες για περιβάλλοντα στα οποία απαιτείται η επαναλαμβανόμενη δημιουργία εικονικών σταθμών με κοινή βασική διαμόρφωση. Οι συγκεκριμένες ρυθμίσεις που εφαρμόστηκαν στο Windows περιβάλλον της παρούσας εργασίας παρουσιάζονται στο Κεφάλαιο 5.

3.4.2 GNU/Linux

Το GNU/Linux αποτελεί τη δεύτερη οικογένεια λειτουργικών συστημάτων που χρησιμοποιείται για τους εικονικούς σταθμούς της εργασίας. Η διαθεσιμότητα διαφορετικών διανομών και ενός εκτεταμένου συνόλου ελεύθερου και ανοικτού λογισμικού το καθιστά ιδιαίτερα κατάλληλο για εκπαιδευτικά και εργαστηριακά περιβάλλοντα στα οποία χρησιμοποιούνται εργαλεία ανάπτυξης, προγραμματισμού και διαχείρισης συστημάτων.

Οι διανομές GNU/Linux υποστηρίζονται άμεσα ως φιλοξενούμενα λειτουργικά συστήματα σε περιβάλλοντα QEMU/KVM. Όπως και στην περίπτωση των Windows, μπορούν να αξιοποιούν παραεικονικοποιημένες συσκευές VirtIO και το QEMU Guest Agent, επιτρέποντας την αποδοτική λειτουργία της εικονικής μηχανής και την επικοινωνία με το σύστημα φιλοξενίας [32], [36].

Για την παροχή γραφικού περιβάλλοντος μέσω απομακρυσμένης πρόσβασης μπορεί να χρησιμοποιηθεί ένας ελαφρύς desktop environment σε συνδυασμό με κατάλληλο RDP server. Η επιλογή ενός σχετικά περιορισμένου γραφικού περιβάλλοντος είναι συμβατή με τον χαρακτήρα ενός εικονικού σταθμού εργασίας, καθώς επιτρέπει να διατίθεται πλήρης γραφική συνεδρία χωρίς να απαιτείται η εγκατάσταση ενός ιδιαίτερα σύνθετου τοπικού περιβάλλοντος. Οι τεχνολογίες XRDP και το γραφικό περιβάλλον που χρησιμοποιήθηκαν στην υλοποίηση εξετάζονται στις επόμενες ενότητες και στο Κεφάλαιο 5.

Η χρήση Linux παράλληλα με Windows επιτρέπει στην ίδια κεντρική υποδομή εικονικοποίησης να παρέχει διαφορετικές κατηγορίες περιβαλλόντων εργασίας. Η δυνατότητα αυτή είναι ιδιαίτερα χρήσιμη σε ένα εκπαιδευτικό πλαίσιο όπου οι απαιτήσεις λογισμικού μπορεί να διαφέρουν μεταξύ μαθημάτων, χωρίς να είναι απαραίτητο οι αντίστοιχες διαμορφώσεις να συνυπάρχουν σε κάθε εικονικό ή φυσικό σταθμό.

3.4.3 Πρότυπα Εικονικών Μηχανών και Κλωνοποίηση

Η δημιουργία πολλών εικονικών μηχανών με παρόμοια χαρακτηριστικά μπορεί να απλοποιηθεί μέσω της χρήσης προτύπων (templates). Ένα πρότυπο αποτελεί μία προετοιμασμένη εικονική μηχανή που χρησιμοποιείται ως βάση για τη δημιουργία νέων VM με κοινή αρχική διαμόρφωση. Με τον τρόπο αυτό, το λειτουργικό σύστημα, οι βασικές εφαρμογές και οι απαραίτητες ρυθμίσεις μπορούν να προετοιμαστούν μία φορά και να επαναχρησιμοποιηθούν για την παραγωγή πολλαπλών εικονικών σταθμών [27], [32].

Στο Proxmox VE, μία κατάλληλα προετοιμασμένη εικονική μηχανή μπορεί να μετατραπεί σε template και στη συνέχεια να χρησιμοποιηθεί ως πηγή κλωνοποίησης. Η πλατφόρμα υποστηρίζει πλήρεις κλώνους (full clones) και, όταν το επιτρέπει η χρησιμοποιούμενη υποδομή αποθήκευσης, συνδεδεμένους κλώνους (linked clones) [32].

Ένας πλήρης κλώνος διαθέτει ανεξάρτητο αντίγραφο των αντίστοιχων εικονικών δίσκων και

μπορεί να χρησιμοποιηθεί ανεξάρτητα από το αρχικό template. Η δημιουργία του απαιτεί όμως την αντιγραφή των δεδομένων και συνεπώς μεγαλύτερη κατανάλωση αποθηκευτικού χώρου και χρόνο δημιουργίας.

Ένας συνδεδεμένος κλώνος βασίζεται σε μία κοινή βασική εικόνα και αποθηκεύει κυρίως τις μεταβολές που πραγματοποιούνται σε σχέση με αυτή. Η προσέγγιση αυτή μπορεί να μειώσει σημαντικά τον απαιτούμενο χώρο και τον χρόνο δημιουργίας νέων VM, αλλά δημιουργεί εξάρτηση του κλώνου από το αρχικό template και από τις δυνατότητες της χρησιμοποιούμενης υποδομής αποθήκευσης [32].

Τα χαρακτηριστικά αυτά καθιστούν τα templates και την κλωνοποίηση ιδιαίτερα χρήσιμα σε περιβάλλοντα VDI, όπου πολλοί εικονικοί σταθμοί μπορούν να προέρχονται από μικρό αριθμό κοινών βασικών εικόνων. Αντί να εγκαθίσταται και να διαμορφώνεται ανεξάρτητα κάθε VM, οι βασικές ιδιότητες του περιβάλλοντος καθορίζονται στο πρότυπο και οι απαιτούμενοι σταθμοί παράγονται από αυτό.

Η χρήση προτύπων δεν εξαλείφει την ανάγκη κατάλληλης προετοιμασίας του φιλοξενούμενου λειτουργικού συστήματος. Ιδιαίτερα στην περίπτωση των Windows, η βασική εικόνα πρέπει να έχει προετοιμαστεί κατάλληλα πριν χρησιμοποιηθεί για επαναλαμβανόμενη ανάπτυξη. Αντίστοιχα, σε ένα Linux περιβάλλον πρέπει να έχουν εγκατασταθεί και διαμορφωθεί οι υπηρεσίες που απαιτούνται από τον τελικό εικονικό σταθμό. Οι συγκεκριμένες διαδικασίες δημιουργίας των βασικών εικόνων και των τελικών προτύπων της παρούσας εργασίας περιγράφονται στο Κεφάλαιο 5.

3.5 Remote Desktop Protocol και FreeRDP

3.5.1 RDP στην Πρακτική Υλοποίηση VDI

Όπως παρουσιάστηκε στο προηγούμενο κεφάλαιο, το Remote Desktop Protocol (RDP) επιτρέπει την απομακρυσμένη αλληλεπίδραση με ένα γραφικό περιβάλλον εργασίας μέσω δικτύου. Στην παρούσα εργασία το RDP επιλέχθηκε ως κοινός μηχανισμός πρόσβασης στους εικονικούς σταθμούς, ώστε ο thin client να μπορεί να χρησιμοποιεί την ίδια βασική τεχνολογία σύνδεσης ανεξάρτητα από το αν το απομακρυσμένο λειτουργικό σύστημα είναι Windows ή GNU/Linux. Στα συστήματα Windows η δυνατότητα απομακρυσμένης επιφάνειας εργασίας παρέχεται από τις αντίστοιχες υπηρεσίες του λειτουργικού συστήματος. Στην πλευρά του Linux χρησιμοποιείται συμβατός RDP server, ώστε και τα δύο είδη εικονικών σταθμών να είναι προσβάσιμα μέσω RDP. Η προσέγγιση αυτή περιορίζει την ανάγκη υποστήριξης διαφορετικών πρωτοκόλλων από το τερματικό και επιτρέπει την ύπαρξη κοινού μηχανισμού σύνδεσης για τις δύο κατηγορίες περιβαλλόντων.

Το RDP λειτουργεί στο σύστημα ως το επίπεδο επικοινωνίας μεταξύ του thin client και της εικονικής επιφάνειας εργασίας που έχει ήδη επιλεγεί για τον χρήστη. Δεν αναλαμβάνει την επιλογή της κατάλληλης εικονικής μηχανής ούτε τη διαχείριση του κύκλου ζωής της. Οι λειτουργίες αυτές αποτελούν αρμοδιότητα διαφορετικών στοιχείων της υποδομής, ενώ το RDP χρησιμοποιείται μετά την κατανομή του κατάλληλου σταθμού για την πραγματοποίηση της γραφικής συνεδρίας.

3.5.2 FreeRDP

Το FreeRDP αποτελεί ελεύθερη και ανοικτού κώδικα υλοποίηση του Remote Desktop Protocol και παρέχει εργαλεία πελάτη για τη σύνδεση με απομακρυσμένες υπηρεσίες RDP [40]. Υποστηρίζει διαφορετικά λειτουργικά συστήματα και μπορεί να χρησιμοποιηθεί σε περιβάλλοντα GNU/Linux για την πρόσβαση σε απομακρυσμένες επιφάνειες εργασίας.

Ιδιαίτερο ενδιαφέρον για την παρούσα εργασία παρουσιάζει ο πελάτης γραμμής εντολών `xfreerdp`. Η λειτουργία του μπορεί να καθοριστεί μέσω παραμέτρων κατά την εκκίνησή του, επιτρέποντας να προσδιοριστούν στοιχεία όπως η διεύθυνση του απομακρυσμένου συστήματος, τα στοιχεία σύνδεσης και οι επιλογές παρουσίασης της συνεδρίας. Η δυνατότητα αυτή διευκολύνει την ενσωμάτωση του RDP client σε αυτοματοποιημένες διαδικασίες, καθώς η σύνδεση μπορεί να ξεκινήσει από script χωρίς να απαιτείται προηγούμενη αλληλεπίδραση με ξεχωριστό γραφικό πρόγραμμα.

Η χρήση του FreeRDP επιτρέπει επομένως σε ένα τερματικό βασισμένο σε GNU/Linux να λειτουργεί ως πελάτης για τις εικονικές επιφάνειες εργασίας της υποδομής. Από την οπτική του thin client, η διεύθυνση του κατάλληλου απομακρυσμένου σταθμού μπορεί να παρέχεται από τον μηχανισμό κατανομής και στη συνέχεια να χρησιμοποιείται για την εκκίνηση της αντίστοιχης RDP συνεδρίας. Ο συγκεκριμένος τρόπος με τον οποίο η διαδικασία αυτή αυτοματοποιείται στην τελική υλοποίηση παρουσιάζεται στο Κεφάλαιο 5.

3.5.3 XRDP για Συστήματα Linux

Το XRDP είναι μία ανοικτού κώδικα υλοποίηση εξυπηρετητή RDP για συστήματα που βασίζονται σε Unix και Linux. Επιτρέπει σε συμβατούς RDP clients να συνδέονται σε γραφικές συνεδρίες Linux, παρέχοντας έτσι έναν μηχανισμό απομακρυσμένης πρόσβασης συμβατό με το πρωτόκολλο που χρησιμοποιείται και για τα Windows περιβάλλοντα [41].

Για την παροχή γραφικών συνεδριών μπορεί να χρησιμοποιηθεί το `xorgxrdp`, το οποίο παρέχει τους απαραίτητους μηχανισμούς για τη σύνδεση του XRDP με τον X.Org server [42]. Μέσω αυτού του συνδυασμού, ένας χρήστης που συνδέεται μέσω RDP μπορεί να αποκτήσει γραφική συνεδρία στο Linux guest χωρίς να απαιτείται διαφορετικό πρωτόκολλο στην πλευρά του thin client.

Η χρήση XRDP και `xorgxrdp` επιτρέπει επομένως την ένταξη των Linux εικονικών σταθμών στην ίδια βασική ροή απομακρυσμένης πρόσβασης με τα συστήματα Windows. Οι δύο κατηγορίες guest διαφέρουν ως προς τον τρόπο με τον οποίο παρέχουν εσωτερικά τη γραφική συνεδρία, αλλά από την πλευρά του τερματικού μπορούν να προσπελαύνονται μέσω του ίδιου RDP client.

Η συγκεκριμένη διαμόρφωση του XRDP, του `xorgxrdp` και του γραφικού περιβάλλοντος που χρησιμοποιήθηκε στους Linux εικονικούς σταθμούς αποτελεί μέρος της υλοποίησης και παρουσιάζεται στο Κεφάλαιο 5.

3.6 Thin Client

3.6.1 Debian GNU/Linux ως Βάση του Thin Client

Για την υλοποίηση του thin client επιλέχθηκε το Debian GNU/Linux. Το Debian αποτελεί διανομή ελεύθερου και ανοικτού λογισμικού με έμφαση στη σταθερότητα και παρέχει εκτεταμένο σύστημα διαχείρισης πακέτων μέσω του APT [43]. Τα χαρακτηριστικά αυτά επιτρέπουν τη δημιουργία ενός σχετικά περιορισμένου λειτουργικού περιβάλλοντος στο οποίο εγκαθίστανται μόνο τα στοιχεία που απαιτούνται για τη δικτυακή επικοινωνία και την απομακρυσμένη πρόσβαση.

Στην περίπτωση ενός thin client δεν απαιτείται το τοπικό σύστημα να παρέχει το σύνολο των εφαρμογών που χρησιμοποιεί ο τελικός χρήστης, καθώς αυτές εκτελούνται στην απομακρυσμένη εικονική μηχανή. Το τοπικό λειτουργικό σύστημα πρέπει κυρίως να υποστηρίζει το απαραίτητο γραφικό περιβάλλον, τη δικτυακή σύνδεση και τον RDP client. Η χρήση μιας διανομής GNU/Linux επιτρέπει τα στοιχεία αυτά να συνδυαστούν με εργαλεία κελύφους και άλλους τυπικούς μηχανισμούς του λειτουργικού συστήματος για την αυτοματοποίηση της διαδικασίας σύνδεσης.

Η επιλογή του Debian είναι επομένως συμβατή με τον περιορισμένο ρόλο που έχει το τερματικό στη συνολική αρχιτεκτονική. Ο thin client δεν αποτελεί το σύστημα στο οποίο εκτελείται το κύριο περιβάλλον εργασίας του χρήστη, αλλά λειτουργεί ως σημείο πρόσβασης προς τους ειδικούς σταθμούς που παρέχονται από την κεντρική υποδομή. Η συγκεκριμένη διαμόρφωση του Debian συστήματος που χρησιμοποιήθηκε στην πειραματική υλοποίηση παρουσιάζεται στο Κεφάλαιο 5.

3.6.2 X Window System και Γραφικό Περιβάλλον

Για την εκτέλεση ενός γραφικού RDP client σε σύστημα GNU/Linux απαιτείται κατάλληλο γραφικό περιβάλλον. Στην υλοποίηση του thin client χρησιμοποιούνται οι μηχανισμοί του X Window System, το οποίο παρέχει τη βασική υποδομή για την παρουσίαση γραφικών εφαρμογών και τη διαχείριση εισόδου από συσκευές όπως το πληκτρολόγιο και το ποντίκι [44].

Για έναν σταθμό που χρησιμοποιείται κυρίως για την παρουσίαση μιας απομακρυσμένης επιφάνειας εργασίας δεν είναι απαραίτητο να εκτελείται ένα πλήρες τοπικό desktop environment. Ένα περιορισμένο γραφικό περιβάλλον μπορεί να παρέχει τις λειτουργίες που απαιτούνται για την εκτέλεση του RDP client, μειώνοντας παράλληλα την πολυπλοκότητα του τοπικού συστήματος.

Η εκκίνηση μιας συνεδρίας X μπορεί να πραγματοποιηθεί μέσω του `startx`, το οποίο χρησιμοποιεί το `xinit` για την εκκίνηση του X server και των εφαρμογών που έχουν καθοριστεί για τη συνεδρία. Μέσω ενός αρχείου όπως το `.xinitrc` μπορεί να οριστεί ποια εφαρμογή ή διαδικασία θα ξεκινήσει όταν δημιουργηθεί η γραφική συνεδρία. Η δυνατότητα αυτή είναι κατάλληλη για έναν thin client, καθώς επιτρέπει η τοπική γραφική συνεδρία να προσανατολίζεται άμεσα στην εκτέλεση του λογισμικού απομακρυσμένης πρόσβασης αντί στην παροχή ενός γενικού περιβάλλοντος εργασίας[45].

3.6.3 Αυτοματοποιημένη Εκκίνηση και Σύνδεση

Ένας thin client μπορεί να απλοποιήσει σημαντικά την αλληλεπίδραση του τελικού χρήστη όταν οι λειτουργίες που απαιτούνται πριν από την απομακρυσμένη σύνδεση εκτελούνται αυτόματα. Αντί ο χρήστης να πραγματοποιεί χειροκίνητη είσοδο στο τοπικό σύστημα, να εκκινεί το γραφικό περιβάλλον και να εισάγει τη διεύθυνση του απομακρυσμένου σταθμού, τα επιμέρους βήματα μπορούν να οργανωθούν σε μία προκαθορισμένη ακολουθία.

Τα συστήματα GNU/Linux παρέχουν διαφορετικούς μηχανισμούς για την αυτοματοποίηση της διαδικασίας εκκίνησης και της έναρξης εφαρμογών μετά την είσοδο ενός χρήστη. Σε συνδυασμό με scripts κελύφους, οι μηχανισμοί αυτοί επιτρέπουν την εκτέλεση μιας ελεγχόμενης ροής στην οποία το τερματικό μπορεί να λαμβάνει τις απαραίτητες πληροφορίες σύνδεσης και στη συνέχεια να εκκινεί αυτόματα τον RDP client.

Η προσέγγιση αυτή είναι κατάλληλη για ένα τερματικό ειδικού σκοπού, καθώς περιορίζει τις ενέργειες που πρέπει να γνωρίζει ή να εκτελεί ο τελικός χρήστης και επιτρέπει η τοπική λειτουργία του σταθμού να επικεντρώνεται στην πρόσβαση προς την κεντρική υποδομή. Ταυτόχρονα, η λογική επιλογής του κατάλληλου εικονικού σταθμού μπορεί να παραμένει ανεξάρτητη από το τοπικό γραφικό περιβάλλον και να εκτελείται από διαφορετικό στοιχείο του συστήματος.

Ο συγκεκριμένος μηχανισμός αυτόματης εισόδου, επιλογής περιβάλλοντος, εκκίνησης της γραφικής συνεδρίας και σύνδεσης με τον κατανεμημένο εικονικό σταθμό αποτελεί μέρος της υλοποίησης και παρουσιάζεται αναλυτικά στο Κεφάλαιο 5.

3.7 SSH και Απομακρυσμένη Εκτέλεση Εντολών

3.7.1 OpenSSH

Το Secure Shell (SSH) αποτελεί πρωτόκολλο για ασφαλή απομακρυσμένη επικοινωνία και χρησιμοποιείται ευρέως για την πρόσβαση και διαχείριση συστημάτων μέσω δικτύου. Παρέχει κρυπτογραφημένο κανάλι επικοινωνίας και υποστηρίζει, μεταξύ άλλων, απομακρυσμένη εκτέλεση εντολών, μεταφορά δεδομένων και διαφορετικούς μηχανισμούς αυθεντικοποίησης [46].

Το OpenSSH αποτελεί ανοικτού κώδικα υλοποίηση του SSH και περιλαμβάνει τόσο τον πελάτη `ssh` όσο και τον εξυπηρετητή `sshd`, μαζί με βοηθητικά εργαλεία για τη δημιουργία και διαχείριση κλειδιών [47]. Η διαθεσιμότητά του σε συστήματα GNU/Linux και η δυνατότητα χρήσης του από τη γραμμή εντολών το καθιστούν κατάλληλο για επικοινωνία που πρέπει να ενσωματωθεί σε αυτοματοποιημένες διαδικασίες.

Για τις ανάγκες ενός αυτοματοποιημένου συστήματος, το SSH δεν είναι απαραίτητο να χρησιμοποιείται ως γενικού σκοπού απομακρυσμένο κέλυφος. Μία σύνδεση μπορεί να χρησιμοποιείται αποκλειστικά για την ενεργοποίηση μιας συγκεκριμένης λειτουργίας στην απομακρυσμένη πλευρά, επιτρέποντας στο SSH να λειτουργεί ως ασφαλές κανάλι μεταφοράς ενός περιορισμένου αιτήματος μεταξύ δύο στοιχείων της υποδομής.

3.7.2 Αυθεντικοποίηση και Περιορισμός Πρόσβασης

Το SSH υποστηρίζει αυθεντικοποίηση με ζεύγη δημόσιου και ιδιωτικού κλειδιού. Στην περίπτωση αυτή, το ιδιωτικό κλειδί παραμένει στην πλευρά του πελάτη, ενώ το αντίστοιχο δημόσιο κλειδί καταχωρίζεται στον απομακρυσμένο λογαριασμό. Η προσέγγιση αυτή επιτρέπει την πραγματοποίηση μη διαδραστικών συνδέσεων χωρίς να απαιτείται η εισαγωγή κωδικού πρόσβασης σε κάθε αίτημα, χαρακτηριστικό ιδιαίτερα χρήσιμο για αυτοματοποιημένες διαδικασίες [48].

Η αποδοχή ενός δημόσιου κλειδιού δεν είναι απαραίτητο να παρέχει στον κάτοχό του όλες τις δυνατότητες μιας συμβατικής SSH συνεδρίας. Το αρχείο `authorized_keys` επιτρέπει τη σύνδεση ενός κλειδιού με πρόσθετους περιορισμούς, μέσω των οποίων μπορούν να απενεργοποιηθούν λειτουργίες όπως η προώθηση θυρών, η προώθηση του SSH agent, η προώθηση X11 ή η δημιουργία ψευδοτερματικού[48].

Οι περιορισμοί αυτοί είναι σημαντικοί όταν μία SSH ταυτότητα χρησιμοποιείται από αυτοματοποιημένο τερματικό και όχι από διαχειριστή. Σε μια τέτοια περίπτωση, ο στόχος δεν είναι να παρέχεται γενική πρόσβαση στον απομακρυσμένο host, αλλά μόνο η ελάχιστη λειτουργικότητα που απαιτείται για την υποβολή του συγκεκριμένου αιτήματος. Με αυτόν τον τρόπο, το SSH μπορεί να χρησιμοποιηθεί ως ελεγχόμενο σημείο εισόδου προς μία υπηρεσία χωρίς να μετατρέπεται αυτομάτως σε πλήρη απομακρυσμένη συνεδρία κελύφους.

Η προσέγγιση αυτή συνδέεται με την αρχή του ελάχιστου προνομίου, σύμφωνα με την οποία ένας λογαριασμός ή μία διεργασία πρέπει να διαθέτει μόνο τα δικαιώματα που απαιτούνται για την εκτέλεση της προβλεπόμενης λειτουργίας [49]. Η εφαρμογή της αρχής αυτής είναι ιδιαίτερα σημαντική όταν ένα σύστημα εκθέτει αυτοματοποιημένες λειτουργίες που τελικά μπορούν να προκαλέσουν ενέργειες διαχείρισης στην υποδομή.

3.7.3 Forced Commands και Αυτοματοποιημένη Επικοινωνία

Το OpenSSH επιτρέπει τον καθορισμό προκαθορισμένης εντολής για ένα συγκεκριμένο δημόσιο κλειδί μέσω της επιλογής `command` στο αρχείο `authorized_keys` [48]. Όταν χρησιμοποιείται αυτή η δυνατότητα, μία επιτυχής σύνδεση με το συγκεκριμένο κλειδί δεν οδηγεί σε αυθαίρετη εκτέλεση εντολών από τον πελάτη. Αντίθετα, ο SSH server εκτελεί την εντολή που έχει προκαθοριστεί από τον διαχειριστή.

Η λειτουργία αυτή είναι κατάλληλη για την έκθεση μιας περιορισμένης υπηρεσίας μέσω SSH. Ένα απομακρυσμένο σύστημα μπορεί να πραγματοποιεί σύνδεση και να ενεργοποιεί μία συγκεκριμένη διαδικασία χωρίς να αποκτά γενικό κέλυφος στον εξυπηρετητή. Η προκαθορισμένη εντολή μπορεί στη συνέχεια να αναλάβει την επικύρωση του αιτήματος, την εκτέλεση των απαιτούμενων ενεργειών και την επιστροφή μόνο του αποτελέσματος που χρειάζεται ο πελάτης.

Σε ένα αυτοματοποιημένο σύστημα διάθεσης εικονικών σταθμών, ο μηχανισμός αυτός επιτρέπει τον σαφή διαχωρισμό μεταξύ της υποβολής ενός αιτήματος και των προνομιούχων λειτουργιών που απαιτούνται για την εξυπηρέτησή του. Ο thin client μπορεί να χρησιμοποιεί μία περιορισμένη SSH ταυτότητα για να ζητήσει έναν εικονικό σταθμό, χωρίς να απαιτείται να διαθέτει άμεση πρόσβαση στις εντολές διαχείρισης της πλατφόρμας εικονικοποίησης.

Στην παρούσα εργασία, το OpenSSH αξιοποιείται με αυτή τη λογική ως σημείο εισόδου προς τον μηχανισμό κατανομής. Η σύνδεση του thin client αντιστοιχίζεται σε προκαθορισμένη λει-

τουργία και ο λογαριασμός εισόδου παραμένει διαχωρισμένος από την ταυτότητα που εκτελεί τις απαιτούμενες λειτουργίες κατανομής. Η ακριβής διαμόρφωση του `authorized_keys`, η αλυσίδα εκτέλεσης και οι κανόνες ανάθεσης δικαιωμάτων παρουσιάζονται στο Κεφάλαιο 5.

3.8 Bash και Αυτοματοποίηση

3.8.1 Shell Scripting

Το Bash (Bourne Again Shell) αποτελεί κέλυφος γραμμής εντολών και γλώσσα scripting που χρησιμοποιείται ευρέως σε συστήματα GNU/Linux. Εκτός από τη διαδραστική εκτέλεση εντολών, επιτρέπει τη σύνθεση πολλαπλών εργαλείων σε scripts, υποστηρίζοντας μεταβλητές, συνθήκες, επαναλήψεις, συναρτήσεις, χειρισμό εισόδου και εξόδου και έλεγχο των κωδικών επιστροφής των εκτελούμενων προγραμμάτων [50].

Η χρήση shell scripts είναι ιδιαίτερα κατάλληλη όταν οι απαιτούμενες λειτουργίες μπορούν να εκφραστούν ως συνδυασμός υπαρχόντων εργαλείων γραμμής εντολών. Αντί να υλοποιούνται εκ νέου λειτουργίες που ήδη παρέχονται από το λειτουργικό σύστημα ή από εξωτερικά προγράμματα, ένα script μπορεί να συντονίζει την εκτέλεσή τους, να ελέγχει τα αποτελέσματα και να καθορίζει την επόμενη ενέργεια ανάλογα με την κατάσταση του συστήματος.

Η δυνατότητα αυτή είναι χρήσιμη στην αυτοματοποίηση υποδομών, όπου πολλές λειτουργίες διαχείρισης παρέχονται ήδη μέσω εργαλείων γραμμής εντολών. Στην περίπτωση του Proxmox VE, για παράδειγμα, λειτουργίες πάνω στις εικονικές μηχανές μπορούν να εκτελούνται μέσω του `qm`, ενώ άλλα εργαλεία του Linux μπορούν να χρησιμοποιούνται για διαχείριση αρχείων, συγχρονισμό διεργασιών και καταγραφή συμβάντων. Το Bash μπορεί να λειτουργεί ως το επίπεδο ενορχήστρωσης που συνδέει αυτές τις επιμέρους δυνατότητες σε επαναλήψιμες διαδικασίες.

Στην παρούσα εργασία τα shell scripts χρησιμοποιούνται για την αυτοματοποίηση βασικών λειτουργιών του κύκλου ζωής των εικονικών σταθμών, όπως η κατανομή διαθέσιμων VM, η δημιουργία και συμπλήρωση των δεξαμενών, η ελεγχόμενη επαναφορά τους σε προκαθορισμένη κατάσταση, η κατάργηση διαχειριζόμενων εικονικών μηχανών και η παρουσίαση της κατάστασης της υποδομής. Η συγκεκριμένη λογική και η υλοποίηση των scripts παρουσιάζονται στο Κεφάλαιο 5.

3.8.2 Διαχείριση Ταυτόχρονων Αιτημάτων

Η αυτοματοποίηση λειτουργιών πάνω σε κοινόχρηστους πόρους δημιουργεί την ανάγκη συντονισμού μεταξύ διεργασιών που ενδέχεται να εκτελούνται ταυτόχρονα. Εάν δύο ανεξάρτητες διεργασίες ελέγξουν και μεταβάλουν τον ίδιο πόρο χωρίς κατάλληλο συγχρονισμό, μπορεί να δημιουργηθούν συνθήκες ανταγωνισμού (race conditions), στις οποίες το τελικό αποτέλεσμα εξαρτάται από τη χρονική σειρά εκτέλεσης των επιμέρους ενεργειών.

Το πρόβλημα είναι ιδιαίτερα σημαντικό σε έναν μηχανισμό κατανομής πόρων. Εάν δύο αιτήματα εντοπίσουν ταυτόχρονα τον ίδιο διαθέσιμο εικονικό σταθμό και επιχειρήσουν να τον δεσμεύσουν, υπάρχει κίνδυνος ο ίδιος πόρος να αποδοθεί σε περισσότερους από έναν χρήστες. Αντίστοιχη ανάγκη συντονισμού εμφανίζεται όταν διαφορετικές λειτουργίες διαχείρισης μπορούν να ενεργήσουν πάνω στην ίδια εικονική μηχανή.

Σε συστήματα Linux, ένας τρόπος αντιμετώπισης τέτοιων περιπτώσεων είναι η χρήση advisory file locks. Το εργαλείο `flock` παρέχει πρόσβαση στον αντίστοιχο μηχανισμό κλειδώματος και μπορεί να χρησιμοποιηθεί από shell scripts ώστε μία κρίσιμη περιοχή να εκτελείται μόνο από τη διεργασία που έχει αποκτήσει επιτυχώς το σχετικό lock [51].

Η χρήση κλειδώματος επιτρέπει επομένως σε πολλαπλές διεργασίες να ανταγωνίζονται με ελεγχόμενο τρόπο για έναν κοινό πόρο. Στην περίπτωση μιας δεξαμενής εικονικών σταθμών, η αντιστοίχιση ανεξάρτητου lock σε κάθε VM επιτρέπει σε διαφορετικές διεργασίες να εξετάζουν διαφορετικούς σταθμούς παράλληλα, ενώ αποτρέπεται η ταυτόχρονη διαχείριση του ίδιου σταθμού από περισσότερες από μία διεργασίες. Η συγκεκριμένη εφαρμογή του μηχανισμού κλειδώματος στην υλοποίηση αναλύεται στα επόμενα κεφάλαια.

3.8.3 Αρχεία Ρυθμίσεων και Καταγραφή Συμβάντων

Η απομάκρυνση μεταβλητών παραμέτρων από τον κώδικα των scripts διευκολύνει τη διαχείριση και τροποποίηση ενός αυτοματοποιημένου συστήματος. Τιμές όπως αναγνωριστικά προτύπων, επιθυμητό μέγεθος δεξαμενών, επιτρεπόμενα εύρη αναγνωριστικών και ονόματα κοινών καταστάσεων μπορούν να αποθηκεύονται σε ξεχωριστά αρχεία ρυθμίσεων αντί να ενσωματώνονται επανειλημμένα στον κώδικα κάθε εργαλείου.

Αντίστοιχα, η κατάσταση ενός διαχειριζόμενου συνόλου πόρων μπορεί να αναπαρίσταται μέσω απλών αρχείων που χρησιμοποιούνται από τα εργαλεία αυτοματοποίησης. Σε ένα σύστημα με διαφορετικές κατηγορίες εικονικών σταθμών, η διατήρηση ξεχωριστής πληροφορίας για τα μέλη κάθε δεξαμενής επιτρέπει στα scripts να περιορίζουν τις ενέργειές τους στους πόρους που ανήκουν στην αντίστοιχη διαχειριζόμενη ομάδα.

Εξίσου σημαντική είναι η καταγραφή των ενεργειών που εκτελούνται από τις αυτοματοποιημένες διαδικασίες. Σε αντίθεση με μία χειροκίνητη ενέργεια, ένα script μπορεί να εκτελεστεί χωρίς άμεση παρακολούθηση από τον διαχειριστή. Η διατήρηση αρχείων καταγραφής ή η αξιοποίηση της υποδομής καταγραφής του λειτουργικού συστήματος επιτρέπει την εκ των υστέρων εξέταση των ενεργειών, των αποτυχιών και των αποτελεσμάτων τους.

Ο συνδυασμός εξωτερικών ρυθμίσεων, σαφούς αναπαράστασης της συμμετοχής των VM στις διαχειριζόμενες δεξαμενές και συστηματικής καταγραφής συμβάντων συμβάλλει στη δημιουργία αυτοματισμών που μπορούν να ελεγχθούν και να συντηρηθούν ευκολότερα. Στην τελική υλοποίηση της εργασίας οι αρχές αυτές εφαρμόζονται σε κοινές ρυθμίσεις κύκλου ζωής, αρχεία δεξαμενών και μηχανισμούς καταγραφής, οι οποίοι παρουσιάζονται αναλυτικά στο Κεφάλαιο 5.

3.9 Σύνοψη Κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκαν οι βασικές τεχνολογίες και τα εργαλεία που επιλέχθηκαν για την ανάπτυξη του συστήματος. Το Proxmox VE και οι τεχνολογίες QEMU/KVM παρέχουν την υποδομή εκτέλεσης και διαχείρισης των εικονικών μηχανών, ενώ το QEMU Guest Agent επιτρέπει την επικοινωνία με τα φιλοξενούμενα λειτουργικά συστήματα. Οι εικονικοί σταθμοί βασίζονται σε περιβάλλοντα Windows και GNU/Linux και μπορούν να δημιουργούνται από κοινά πρότυπα μέσω των μηχανισμών κλωνοποίησης της πλατφόρμας. Η απομακρυσμένη πρόσβαση πραγματοποιείται μέσω RDP, με FreeRDP στην πλευρά του thin client και

XRDP για την παροχή αντίστοιχης υπηρεσίας από τους σταθμούς Linux.

Παράλληλα, το Debian GNU/Linux παρέχει τη βάση για την υλοποίηση του thin client, ενώ το OpenSSH και τα scripts Bash υποστηρίζουν την ελεγχόμενη επικοινωνία και την αυτοματοποίηση των λειτουργιών διαχείρισης. Οι τεχνολογίες αυτές δεν λειτουργούν ως ανεξάρτητα στοιχεία, αλλά συνδυάζονται ώστε να σχηματίσουν μία ενιαία υποδομή διάθεσης και διαχείρισης εικονικών σταθμών εργασίας. Ο τρόπος με τον οποίο οργανώνονται σε επιμέρους δομικά στοιχεία, οι μεταξύ τους σχέσεις και οι βασικές ροές λειτουργίας του συστήματος αποτελούν αντικείμενο του επόμενου κεφαλαίου.

Κεφάλαιο 4

Σχεδίαση και Αρχιτεκτονική του Συστήματος

4.1 Εισαγωγή

Μετά την παρουσίαση του θεωρητικού υποβάθρου και των τεχνολογιών που αξιοποιούνται στην παρούσα εργασία, το κεφάλαιο αυτό επικεντρώνεται στη σχεδίαση και την αρχιτεκτονική του προτεινόμενου συστήματος VDI. Στόχος είναι να αποτυπωθεί ο τρόπος με τον οποίο τα επιμέρους δομικά στοιχεία συνδυάζονται σε ένα ενιαίο σύστημα για την κεντρική διάθεση, ανάθεση και διαχείριση εικονικών σταθμών εργασίας σε περιβάλλον εκπαιδευτικού εργαστηρίου.

Η σχεδίαση βασίζεται στη διατήρηση κεντρικά διαχειριζόμενων προτύπων εικονικών μηχανών και στη δημιουργία διακριτών ομάδων διαθέσιμων σταθμών εργασίας για περιβάλλοντα Linux και Windows. Η πρόσβαση των χρηστών πραγματοποιείται μέσω ελαφρών thin clients, ενώ ένας ενδιάμεσος μηχανισμός broker αναλαμβάνει την επιλογή και ανάθεση κατάλληλης εικονικής μηχανής. Παράλληλα, προβλέπονται μηχανισμοί διαχείρισης του κύκλου ζωής των εικονικών μηχανών, ώστε οι σταθμοί να μπορούν να δημιουργούνται, να επαναφέρονται σε προκαθορισμένη καθαρή κατάσταση, να αποσύρονται και να αναπληρώνονται με ελεγχόμενο τρόπο. Η συνολική αρχιτεκτονική επιδιώκει έτσι να περιορίσει τις επαναλαμβανόμενες χειροκίνητες ενέργειες διαχείρισης και να υποστηρίξει τη μαζική διαχείριση των εργαστηριακών συστημάτων από κεντρικό σημείο.

Στις επόμενες ενότητες παρουσιάζονται αρχικά οι βασικές απαιτήσεις και οι στόχοι σχεδίασης και στη συνέχεια η συνολική αρχιτεκτονική του συστήματος. Ακολουθεί η ανάλυση της οργάνωσης των εικονικών σταθμών εργασίας και των pools, του μηχανισμού broker και της ροής πρόσβασης από τον thin client, καθώς και της σχεδίασης των λειτουργιών διαχείρισης του κύκλου ζωής των εικονικών μηχανών. Τέλος, εξετάζονται οι επιλογές ασφάλειας και διαχωρισμού δικαιωμάτων, καθώς και τα όρια της προτεινόμενης σχεδίασης και οι δυνατότητες μελλοντικής επέκτασής της.

4.2 Απαιτήσεις και Στόχοι Σχεδίασης

Η σχεδίαση του συστήματος καθορίστηκε από τις ανάγκες ενός εκπαιδευτικού εργαστηρίου, στο οποίο απαιτείται η διάθεση πολλαπλών σταθμών εργασίας με όσο το δυνατόν μικρότερη ανάγκη χειροκίνητης διαχείρισης. Βασικός στόχος είναι η μεταφορά του κύριου διαχειριστικού φόρτου από τους επιμέρους φυσικούς υπολογιστές σε ένα κεντρικά ελεγχόμενο περιβάλλον εικονικοποίησης.

Με βάση αυτό το πλαίσιο, οι κύριες απαιτήσεις σχεδίασης συνοψίζονται ως εξής:

- **Κεντρική διαχείριση των σταθμών εργασίας.** Η δημιουργία, η διάθεση και η επαναφορά των εικονικών σταθμών πρέπει να πραγματοποιούνται από κεντρικό σημείο, χωρίς να απαιτείται ανεξάρτητη διαχείριση κάθε τερματικού.
- **Υποστήριξη διαφορετικών λειτουργικών συστημάτων.** Το σύστημα πρέπει να μπορεί να διαθέτει περιβάλλοντα Linux και Windows μέσω διακριτών ομάδων εικονικών μηχανών, ώστε ο χρήστης να επιλέγει το κατάλληλο περιβάλλον εργασίας χωρίς να μεταβάλλεται η βασική διαδικασία πρόσβασης.
- **Αυτοματοποιημένη ανάθεση διαθέσιμου σταθμού.** Ο χρήστης δεν πρέπει να χρειάζεται να γνωρίζει τη διεύθυνση ή την ταυτότητα συγκεκριμένης εικονικής μηχανής. Η επιλογή κατάλληλου διαθέσιμου μέλους του αντίστοιχου pool πρέπει να πραγματοποιείται από κεντρικό μηχανισμό ανάθεσης.
- **Επαναχρησιμοποιήσιμη και ομοιόμορφη αρχική κατάσταση.** Οι εικονικοί σταθμοί πρέπει να προέρχονται από προκαθορισμένα πρότυπα και να μπορούν να επιστρέφουν σε γνωστή καθαρή κατάσταση. Με αυτόν τον τρόπο περιορίζεται η συσσώρευση αλλαγών που προκαλούνται κατά τη χρήση τους και διατηρείται συνεπές περιβάλλον μεταξύ διαφορετικών συνεδριών.
- **Διαχείριση της χωρητικότητας των pools.** Η αρχιτεκτονική πρέπει να επιτρέπει τον καθορισμό επιθυμητού αριθμού διαθέσιμων εικονικών σταθμών και την ελεγχόμενη δημιουργία νέων μελών όταν η πραγματική χωρητικότητα υπολείπεται του καθορισμένου στόχου.
- **Ασφαλής διαχείριση του κύκλου ζωής.** Οι λειτουργίες επαναφοράς και καταστροφής πρέπει να περιορίζονται στα αντικείμενα που ανήκουν στο διαχειριζόμενο VDI περιβάλλον και να προστατεύονται από συγκρούσεις με άλλες ταυτόχρονες λειτουργίες. Επιπλέον, η αυτόματη επαναφορά πρέπει να αποφεύγει την επέμβαση σε εικονικές μηχανές που εξυπηρετούν ενεργή συνεδρία χρήστη.
- **Απλοποιημένη εμπειρία χρήστη στον thin client.** Το τερματικό πρόσβασης πρέπει να απαιτεί ελάχιστη αλληλεπίδραση από τον χρήστη. Η επιλογή του επιθυμητού λειτουργικού περιβάλλοντος και η σύνδεση στον αντίστοιχο εικονικό σταθμό πρέπει να πραγματοποιούνται χωρίς να εκτίθενται στον χρήστη οι εσωτερικές διαδικασίες ανάθεσης και διαχείρισης.
- **Περιορισμός δικαιωμάτων και διαχωρισμός ρόλων.** Η πρόσβαση του thin client προς το κεντρικό σύστημα δεν πρέπει να παρέχει γενική διαχειριστική πρόσβαση στον

hypervisor. Κάθε επιμέρους μηχανισμός πρέπει να διαθέτει μόνο τα δικαιώματα που απαιτούνται για τον συγκεκριμένο ρόλο του.

- **Αποφυγή εξαρτήσεων από πολύπλοκες πλατφόρμες διαχείρισης.** Για το συγκεκριμένο πειραματικό περιβάλλον επιδιώκεται μια σχετικά απλή και κατανοητή αρχιτεκτονική, βασισμένη στις δυνατότητες της πλατφόρμας εικονικοποίησης και σε μικρούς αυτοματοποιημένους μηχανισμούς, χωρίς την εισαγωγή πρόσθετου συστήματος orchestration.

Οι απαιτήσεις αυτές οδηγούν σε μια αρχιτεκτονική στην οποία ο hypervisor αποτελεί το κεντρικό σημείο φιλοξενίας και διαχείρισης των εικονικών σταθμών, οι εικονικές μηχανές οργανώνονται σε διακριτά pools, ένας broker μεσολαβεί μεταξύ των thin clients και των διαθέσιμων πόρων, και ανεξάρτητες λειτουργίες διαχείρισης αναλαμβάνουν τον κύκλο ζωής των VDI μελών. Η διάκριση των ρόλων αυτών επιτρέπει την αυτοματοποίηση των συχνότερων λειτουργιών χωρίς να απαιτείται η παροχή ευρείας διαχειριστικής πρόσβασης στους τελικούς χρήστες.

4.3 Συνολική Αρχιτεκτονική του Συστήματος

Η αρχιτεκτονική του συστήματος οργανώνεται γύρω από ένα κεντρικό περιβάλλον εικονικοποίησης, στο οποίο φιλοξενούνται τα πρότυπα και οι εικονικοί σταθμοί εργασίας, ενώ οι τελικοί χρήστες αλληλεπιδρούν με το σύστημα μέσω thin clients. Ανάμεσα στα δύο επίπεδα παρεμβάλλεται ο broker, ο οποίος αναλαμβάνει την αντιστοίχιση ενός αιτήματος πρόσβασης με έναν διαθέσιμο εικονικό σταθμό του κατάλληλου λειτουργικού περιβάλλοντος.

Σε υψηλό επίπεδο, το σύστημα αποτελείται από τέσσερα βασικά λειτουργικά επίπεδα:

1. **Επίπεδο πρόσβασης χρήστη.** Οι thin clients αποτελούν το σημείο αλληλεπίδρασης με τον τελικό χρήστη. Ο χρήστης επιλέγει το επιθυμητό περιβάλλον Linux ή Windows και το τερματικό αναλαμβάνει την υποβολή του αντίστοιχου αιτήματος προς το κεντρικό σύστημα. Μετά την ολοκλήρωση της ανάθεσης, ο thin client συνδέεται απευθείας στον εικονικό σταθμό μέσω του κατάλληλου πρωτοκόλλου απομακρυσμένης επιφάνειας εργασίας.
2. **Επίπεδο ανάθεσης πόρων.** Ο broker λειτουργεί ως ενδιάμεσο επίπεδο μεταξύ των thin clients και της υποδομής εικονικοποίησης. Δέχεται ένα περιορισμένο αίτημα που προσδιορίζει το επιθυμητό pool, εντοπίζει διαθέσιμο μέλος του και ενεργοποιεί την εικονική μηχανή όταν αυτό απαιτείται. Στη συνέχεια αποκτά τη διεύθυνση δικτύου του guest και επιστρέφει στον thin client μόνο την πληροφορία που απαιτείται για την έναρξη της απομακρυσμένης σύνδεσης.
3. **Επίπεδο εικονικών σταθμών εργασίας.** Οι εικονικοί σταθμοί οργανώνονται σε διακριτά pools Linux και Windows και δημιουργούνται από αντίστοιχα προετοιμασμένα πρότυπα. Κάθε pool αποτελεί ένα σύνολο εναλλάξιμων σταθμών του ίδιου λειτουργικού περιβάλλοντος, επιτρέποντας στον broker να επιλέγει διαθέσιμο πόρο χωρίς ο τελικός χρήστης να χρειάζεται να γνωρίζει συγκεκριμένο VMID, όνομα ή διεύθυνση IP.

4. **Επίπεδο διαχείρισης κύκλου ζωής.** Ξεχωριστοί διαχειριστικοί μηχανισμοί αναλαμβάνουν τη διατήρηση της επιθυμητής χωρητικότητας των pools, την επαναφορά εικονικών σταθμών σε προκαθορισμένη καθαρή κατάσταση, την ελεγχόμενη απόσυρση διαχειριζόμενων μελών και την παρακολούθηση της κατάστασης του VDI περιβάλλοντος. Οι λειτουργίες αυτές συνεργάζονται με τον μηχανισμό ανάθεσης, ώστε αντικρουόμενες ενέργειες πάνω στην ίδια εικονική μηχανή να αποφεύγονται.

Η βασική ροή πρόσβασης ξεκινά στον thin client. Μετά την επιλογή λειτουργικού περιβάλλοντος, αποστέλλεται στον broker αίτημα για το αντίστοιχο pool. Ο broker εξετάζει τα καταχωρισμένα μέλη του pool και επιλέγει έναν διαθέσιμο εικονικό σταθμό. Εφόσον απαιτείται, η επιλεγμένη εικονική μηχανή εκκινείται και ο broker αναμένει μέχρι να είναι διαθέσιμη μια κατάλληλη διεύθυνση IPv4 μέσω του QEMU Guest Agent. Η διεύθυνση επιστρέφεται στον thin client, ο οποίος στη συνέχεια δημιουργεί τη σύνδεση απομακρυσμένης επιφάνειας εργασίας απευθείας προς την εικονική μηχανή.

Με τον τρόπο αυτό ο broker συμμετέχει στη διαδικασία ανάθεσης, αλλά δεν παρεμβάλλεται στη μεταφορά της ίδιας της γραφικής συνεδρίας. Μετά την ανάθεση, η επικοινωνία της απομακρυσμένης επιφάνειας εργασίας πραγματοποιείται μεταξύ thin client και εικονικού σταθμού. Η διάκριση αυτή διατηρεί τον broker ως μηχανισμό ελέγχου και συντονισμού, χωρίς να τον μετατρέπει σε ενδιάμεσο κόμβο για όλη την κίνηση της συνεδρίας.

Παράλληλα με τη ροή πρόσβασης λειτουργεί ο κύκλος διαχείρισης των εικονικών σταθμών. Τα μέλη των pools δημιουργούνται από προκαθορισμένα πρότυπα και καταχωρίζονται ως διαθέσιμοι πόροι μόνο αφού έχουν φτάσει στην απαιτούμενη αρχική κατάσταση. Μετά τη χρήση τους μπορούν να επανέλθουν σε προκαθορισμένο καθαρό σημείο επαναφοράς και να διατεθούν εκ νέου. Αν ένα διαχειριζόμενο μέλος αποσυρθεί, η χωρητικότητα του pool μπορεί να αποκατασταθεί με τη δημιουργία νέου σταθμού από το αντίστοιχο πρότυπο. Η λογική αυτή διαχωρίζει την ταυτότητα μιας συγκεκριμένης εικονικής μηχανής από την υπηρεσία που παρέχει το pool: το σημαντικό στοιχείο για τον χρήστη είναι η διαθεσιμότητα ενός κατάλληλου περιβάλλοντος και όχι η διατήρηση ενός συγκεκριμένου VM μεταξύ συνεδριών.

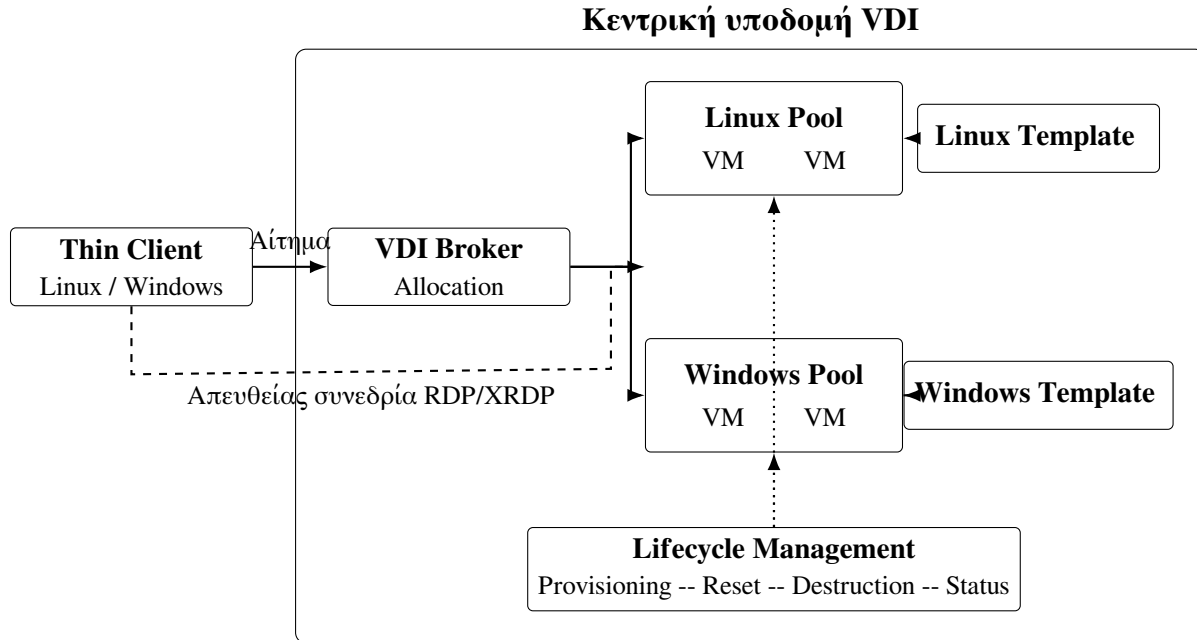
Η συνολική αρχιτεκτονική παρουσιάζεται σχηματικά στο Σχήμα 4.1. Το διάγραμμα αποτυπώνει τόσο τη διαδρομή ενός αιτήματος χρήστη όσο και τη σχέση των μηχανισμών διαχείρισης με τα πρότυπα και τα pools των εικονικών μηχανών.

4.4 Αρχιτεκτονική Εικονικών Σταθμών Εργασίας και Pools

Οι εικονικοί σταθμοί εργασίας αποτελούν τους υπολογιστικούς πόρους που διατίθενται στους τελικούς χρήστες. Αντί να αντιμετωπίζεται κάθε εικονική μηχανή ως ανεξάρτητο και μόνιμο σύστημα, η προτεινόμενη αρχιτεκτονική οργανώνει τους σταθμούς σε ομάδες κοινών χαρακτηριστικών (*pools*) και βασίζεται σε προκαθορισμένα πρότυπα για τη δημιουργία και την ανανέωσή τους.

4.4.1 Πρότυπα και δημιουργία εικονικών σταθμών

Για κάθε υποστηριζόμενο λειτουργικό σύστημα διατηρείται ξεχωριστό πρότυπο εικονικής μηχανής. Τα πρότυπα προέρχονται από προετοιμασμένα συστήματα αναφοράς, στα οποία έχουν



Σχήμα 4.1: Συνολική αρχιτεκτονική του προτεινόμενου συστήματος VDI.

ενσωματωθεί τα απαραίτητα στοιχεία για την απομακρυσμένη πρόσβαση και την επικοινωνία με την υποδομή διαχείρισης. Μετά την ολοκλήρωση της προετοιμασίας τους, χρησιμοποιούνται ως σταθερή βάση για τη δημιουργία των εικονικών σταθμών που εντάσσονται στα αντίστοιχα pools.

Η δημιουργία των μελών πραγματοποιείται με linked clones. Με αυτόν τον τρόπο, ένα κοινό πρότυπο μπορεί να χρησιμοποιηθεί ως βάση για πολλαπλές εικονικές μηχανές χωρίς να απαιτείται η δημιουργία ανεξάρτητου πλήρους αντιγράφου του βασικού δίσκου για κάθε νέο σταθμό[32]. Η επιλογή αυτή είναι συμβατή με τον προσωρινό και επαναχρησιμοποιήσιμο χαρακτήρα των VDI σταθμών της συγκεκριμένης αρχιτεκτονικής, καθώς τα δεδομένα που δημιουργούνται κατά τη χρήση ενός μέλους δεν προορίζονται να μετατρέψουν το ίδιο το πρότυπο.

Η σχέση μεταξύ προτύπου και pool μπορεί επομένως να αποδοθεί ως εξής:

Προετοιμασμένο σύστημα → Template → Linked clones → VDI pool

Το πρότυπο παραμένει εκτός της κανονικής διαδικασίας ανάθεσης και δεν διατίθεται απευθείας στους χρήστες. Οι χρήστες συνδέονται αποκλειστικά σε διαχειριζόμενα αντίγραφα που έχουν ολοκληρώσει την απαιτούμενη διαδικασία προετοιμασίας και έχουν καταχωριστεί στο αντίστοιχο pool.

4.4.2 Διαχωρισμός Linux και Windows pools

Οι εικονικοί σταθμοί Linux και Windows οργανώνονται σε ανεξάρτητα pools. Ο διαχωρισμός αυτός επιτρέπει στο σύστημα να αντιμετωπίζει κάθε λειτουργικό περιβάλλον ως ξεχωριστή κατηγορία πόρων, με δικό της πρότυπο, χώρο αναγνωριστικών, σύνολο καταχωρισμένων μελών και διαδικασία προετοιμασίας.

Ο broker δεν χρειάζεται να γνωρίζει τις εσωτερικές διαφορές προετοιμασίας των δύο λειτουργικών συστημάτων. Λαμβάνει την επιλογή περιβάλλοντος από τον thin client και αναζητά διαθέσιμο σταθμό στο αντίστοιχο pool. Με αυτόν τον τρόπο, η επιλογή Linux ή Windows μεταφράζεται σε επιλογή ομάδας πόρων και όχι σε άμεση επιλογή συγκεκριμένης εικονικής μηχανής. Η αρχιτεκτονική διατηρεί επίσης διακριτά εύρη αναγνωριστικών για τα διαχειριζόμενα μέλη κάθε pool. Ο διαχωρισμός των περιοχών αυτών λειτουργεί ως πρόσθετο όριο μεταξύ των δύο κατηγοριών και επιτρέπει στους μηχανισμούς διαχείρισης να ελέγχουν αν μια εικονική μηχανή ανήκει στον χώρο που προορίζεται για το συγκεκριμένο λειτουργικό σύστημα.

4.4.3 Κατάσταση ετοιμότητας και σημείο επαναφοράς

Η απλή δημιουργία μιας εικονικής μηχανής από το αντίστοιχο πρότυπο δεν θεωρείται από μόνη της επαρκής για να καταστεί ο σταθμός διαθέσιμος στον broker. Κάθε νέο μέλος πρέπει πρώτα να φτάσει σε μια προκαθορισμένη κατάσταση ετοιμότητας.

Για τον σκοπό αυτό, η αρχιτεκτονική χρησιμοποιεί ένα σημείο επαναφοράς που αντιπροσωπεύει την καθαρή κατάσταση από την οποία μπορεί να ξεκινήσει μια νέα συνεδρία. Παράλληλα, χρησιμοποιούνται μεταδεδωμένα που χαρακτηρίζουν την εικονική μηχανή ως διαχειριζόμενο VDI μέλος, προσδιορίζουν το λειτουργικό περιβάλλον στο οποίο ανήκει και δηλώνουν ότι έχει ολοκληρωθεί η απαιτούμενη προετοιμασία.

Η σωστή ακολουθία μπορεί να εκφραστεί σε υψηλό επίπεδο ως:

Clone → Προετοιμασία → Χαρακτηρισμός ως έτοιμο VDI μέλος → Clean-state snapshot → Καταχώριση στο pool

Η καταχώριση στο pool αποτελεί σκόπιμο το τελευταίο στάδιο. Έτσι, μια εικονική μηχανή που δημιουργήθηκε αλλά δεν ολοκλήρωσε επιτυχώς την απαιτούμενη προετοιμασία δεν γίνεται ορατή στον μηχανισμό ανάθεσης και δεν μπορεί να δοθεί σε τελικό χρήστη.

Η σειρά μεταξύ μεταδεδωμένων ετοιμότητας και snapshot είναι επίσης σημαντική. Το snapshot αποτελεί όχι μόνο σημείο επαναφοράς της κατάστασης των δίσκων, αλλά μπορεί να επαναφέρει και στοιχεία της διαμόρφωσης της εικονικής μηχανής[32]. Για τον λόγο αυτό, τα χαρακτηριστικά που προσδιορίζουν το VM ως έγκυρο μέλος του VDI περιβάλλοντος πρέπει να έχουν εφαρμοστεί πριν από τη δημιουργία του καθαρού σημείου επαναφοράς.

4.4.4 Διαφοροποίηση της διαδικασίας προετοιμασίας

Παρότι η λογική των pools είναι κοινή, η διαδικασία με την οποία ένα νέο μέλος φτάνει στην κατάσταση ετοιμότητας διαφέρει μεταξύ Linux και Windows.

Στην περίπτωση του Linux, το πρότυπο βρίσκεται ήδη σε κατάσταση κατάλληλη για τη δημιουργία νέων σταθμών και επομένως το linked clone μπορεί να προχωρήσει άμεσα στα στάδια χαρακτηρισμού, δημιουργίας του καθαρού snapshot και καταχώρισης στο pool.

Στην περίπτωση των Windows απαιτείται ένα επιπλέον στάδιο πρώτης εκκίνησης μετά τη δημιουργία του clone. Η εκκίνηση αυτή επιτρέπει στο γενικευμένο λειτουργικό σύστημα να ολοκληρώσει την προβλεπόμενη αρχική προετοιμασία του. Μόνο μετά την ολοκλήρωση αυτής της διαδικασίας και την επιστροφή της εικονικής μηχανής σε σταματημένη κατάσταση δημιουργείται το καθαρό σημείο επαναφοράς και το μέλος καταχωρίζεται στο Windows pool.

Η διαφοροποίηση αυτή περιορίζεται στο στάδιο προετοιμασίας. Από τη στιγμή που ένα μέλος έχει χαρακτηριστεί ως έτοιμο και έχει καταχωριστεί, η βασική λογική διαχείρισης παραμένει κοινή: ο broker αντιμετωπίζει τα μέλη του επιλεγμένου pool ως διαθέσιμους πόρους που μπορούν να ανατεθούν, να χρησιμοποιηθούν, να επανέλθουν στην καθαρή κατάσταση και να διατεθούν εκ νέου.

4.4.5 Λογική χωρητικότητας των pools

Κάθε pool συνδέεται με έναν επιθυμητό αριθμό διαχειριζόμενων μελών. Η χωρητικότητα αυτή δεν συνδέεται αρχιτεκτονικά με συγκεκριμένο σταθερό αριθμό εικονικών μηχανών, αλλά αποτελεί παράμετρο του περιβάλλοντος.

Η διαχείριση της χωρητικότητας βασίζεται στη σύγκριση του επιθυμητού μεγέθους με τον αριθμό των καταχωρισμένων μελών που εξακολουθούν να υπάρχουν στην υποδομή. Όταν εντοπίζεται έλλειμμα, μπορούν να δημιουργηθούν νέα μέλη από το αντίστοιχο πρότυπο μέχρι να αποκατασταθεί η επιθυμητή χωρητικότητα. Αντίθετα, όταν το pool βρίσκεται ήδη στο απαιτούμενο μέγεθος, δεν απαιτείται δημιουργία πρόσθετων εικονικών μηχανών.

Η προσέγγιση αυτή διαχωρίζει την επιθυμητή κατάσταση του pool από την ταυτότητα των επιμέρους μελών του. Ένα συγκεκριμένο VM μπορεί να αποσυρθεί και να αντικατασταθεί, χωρίς να μεταβάλλεται ο ρόλος του pool ως διαθέσιμου συνόλου σταθμών εργασίας. Το μοντέλο αυτό υποστηρίζει τον προσωρινό χαρακτήρα των VDI σταθμών και διευκολύνει την κεντρική διαχείριση πολλαπλών ομοειδών συστημάτων.

4.5 Σχεδίαση του Broker και Μηχανισμός Ανάθεσης

Ο broker αποτελεί το ενδιάμεσο επίπεδο συντονισμού μεταξύ των thin clients και των εικονικών σταθμών εργασίας. Βασικός του ρόλος είναι να μετατρέπει ένα απλό αίτημα για συγκεκριμένο λειτουργικό περιβάλλον σε ανάθεση ενός διαθέσιμου μέλους του αντίστοιχου pool. Με αυτόν τον τρόπο, ο τελικός χρήστης δεν χρειάζεται να γνωρίζει ποια εικονική μηχανή είναι διαθέσιμη, ποιο είναι το αναγνωριστικό της ή ποια διεύθυνση IP έχει αποκτήσει.

Η σχεδίαση του broker διατηρεί σκόπιμα περιορισμένο το εύρος των αρμοδιοτήτων του. Δεν αποτελεί πλήρες σύστημα διαχείρισης της υποδομής ούτε μεσολαβεί στη γραφική συνεδρία του χρήστη. Αναλαμβάνει αποκλειστικά την αποδοχή ενός έγκυρου αιτήματος για pool, την επιλογή κατάλληλου εικονικού σταθμού, την εκκίνησή του όταν απαιτείται και την επιστροφή της διεύθυνσης δικτύου που χρειάζεται ο thin client για την απευθείας σύνδεση.

4.5.1 Επιλογή pool και διαθέσιμου μέλους

Κάθε αίτημα ανάθεσης περιλαμβάνει την κατηγορία περιβάλλοντος που έχει επιλέξει ο χρήστης, δηλαδή Linux ή Windows. Η πληροφορία αυτή αντιστοιχίζεται στο αντίστοιχο pool και περιορίζει την αναζήτηση αποκλειστικά στα μέλη που έχουν καταχωριστεί σε αυτό.

Η καταχώριση στο pool λειτουργεί ως όριο μεταξύ των εικονικών μηχανών που μπορούν να διατεθούν στους χρήστες και των υπόλοιπων αντικειμένων της υποδομής. Πρότυπα, συστήματα προετοιμασίας και άλλες εικονικές μηχανές δεν συμμετέχουν στη διαδικασία ανάθεσης μόνο και μόνο επειδή βρίσκονται στον ίδιο hypervisor.

Για την ανάθεση εξετάζονται τα καταχωρισμένα μέλη μέχρι να εντοπιστεί κατάλληλη εικονική μηχανή σε σταματημένη κατάσταση. Η εικονική μηχανή αυτή δεσμεύεται για τη συγκεκριμένη διαδικασία ανάθεσης και στη συνέχεια εκκινείται. Αν δεν υπάρχει διαθέσιμο μέλος, το αίτημα δεν πρέπει να οδηγήσει σε αυθαίρετη χρήση άλλου πόρου εκτός του επιλεγμένου pool, αλλά να ολοκληρωθεί με αποτυχία ανάθεσης.

4.5.2 Συγχρονισμός ταυτόχρονων λειτουργιών

Η κατάσταση μιας εικονικής μηχανής δεν αρκεί από μόνη της για ασφαλή επιλογή όταν περισσότερα από ένα αιτήματα μπορούν να εκτελεστούν ταυτόχρονα. Δύο παράλληλες διαδικασίες θα μπορούσαν να παρατηρήσουν το ίδιο VM ως σταματημένο πριν οποιαδήποτε από αυτές προλάβει να το εκκινήσει.

Για τον λόγο αυτό, η αρχιτεκτονική προβλέπει μηχανισμό αμοιβαίου αποκλεισμού ανά εικονική μηχανή. Πριν ένα μέλος επιλεγεί οριστικά, η διαδικασία ανάθεσης πρέπει να αποκτήσει αποκλειστική πρόσβαση σε έναν κοινό μηχανισμό κλειδώματος που αντιστοιχεί στο συγκεκριμένο VM. Η κατάστασή του επανελέγχεται ενώ το κλείδωμα βρίσκεται ήδη στην κατοχή της διαδικασίας και μόνο τότε μπορεί να θεωρηθεί διαθέσιμο.

Η λογική αυτή δεν περιορίζεται στον broker. Το ίδιο πεδίο κλειδώματος χρησιμοποιείται και από τις λειτουργίες του κύκλου ζωής που μπορούν να μεταβάλουν την κατάσταση ενός VM. Έτσι, η ανάθεση, η επαναφορά και η απόσυρση ενός σταθμού δεν επιτρέπεται να εκτελούνται ταυτόχρονα πάνω στο ίδιο μέλος.

Σε αφαιρετική μορφή, η διαδικασία επιλογής μπορεί να αποδοθεί ως:

Pool member → *Lock* → *State re-check* → *Selection*

Το κλείδωμα διατηρείται κατά το κρίσιμο τμήμα της διαδικασίας ανάθεσης, ώστε το επιλεγμένο VM να μη θεωρηθεί ταυτόχρονα διαθέσιμο από άλλη συνεργαζόμενη λειτουργία.

4.5.3 Εκκίνηση και ανακάλυψη διεύθυνσης δικτύου

Μετά την επιλογή του διαθέσιμου μέλους, ο broker αναλαμβάνει την εκκίνησή του. Η μετάβαση όμως της εικονικής μηχανής σε κατάσταση λειτουργίας δεν σημαίνει ότι είναι άμεσα έτοιμη να δεχθεί σύνδεση από τον thin client. Το guest λειτουργικό σύστημα πρέπει πρώτα να ολοκληρώσει την εκκίνησή του και να αποκτήσει χρησιμοποιήσιμη διεύθυνση δικτύου.

Η ανακάλυψη της διεύθυνσης πραγματοποιείται μέσω του QEMU Guest Agent. Ο broker επικοινωνεί με τον guest μέσω του management channel του hypervisor και αναζητά κατάλληλη IPv4 διεύθυνση στις δικτυακές διεπαφές του. Με αυτόν τον τρόπο, η διαδικασία ανάθεσης δεν εξαρτάται από στατικά καθορισμένες διευθύνσεις για κάθε εικονική μηχανή ούτε απαιτεί από τον thin client να ανακαλύπτει μόνος του τον σταθμό που του ανατέθηκε.

Η διαδικασία μπορεί να συνοψιστεί ως:

Επιλογή VM → *Εκκίνηση* → *Αναμονή guest* → *Ανακάλυψη IPv4* → *Επιστροφή στον thin client*

Η ύπαρξη χρονικού ορίου στην αναμονή αποτελεί επίσης απαραίτητο χαρακτηριστικό της σχεδίασης. Ένα VM που αποτυγχάνει να ολοκληρώσει την εκκίνηση ή του οποίου ο Guest Agent δεν παρέχει χρησιμοποιήσιμη δικτυακή πληροφορία δεν πρέπει να δεσμεύει επ' αόριστον τη διαδικασία ανάθεσης.

4.5.4 Διαχωρισμός ανάθεσης και απομακρυσμένης συνεδρίας

Μετά την επιτυχή ανακάλυψη της διεύθυνσης, ο broker επιστρέφει στον thin client μόνο την πληροφορία που απαιτείται για τη συνέχεια της σύνδεσης. Η απομακρυσμένη γραφική συνεδρία δημιουργείται στη συνέχεια απευθείας μεταξύ του thin client και του επιλεγμένου VM. Ο διαχωρισμός αυτός δημιουργεί δύο διαφορετικές ροές:

1. τη **ροή ελέγχου**, μέσω της οποίας ο thin client ζητά από τον broker έναν διαθέσιμο σταθμό και λαμβάνει τη διεύθυνσή του,
2. τη **ροή συνεδρίας**, μέσω της οποίας ο thin client επικοινωνεί απευθείας με τον εικονικό σταθμό μέσω RDP ή XRDP.

Επομένως, ο broker δεν χρειάζεται να μεταφέρει τη συνεχή γραφική κίνηση των ενεργών χρηστών. Ο ρόλος του ολοκληρώνεται ουσιαστικά με την ασφαλή επιλογή του πόρου και την παράδοση της πληροφορίας σύνδεσης στον thin client.

4.5.5 Περιορισμένη διεπαφή προς τον thin client

Η επικοινωνία του thin client με τον broker σχεδιάζεται ως περιορισμένη διεπαφή και όχι ως γενική απομακρυσμένη πρόσβαση στον κεντρικό εξυπηρετητή. Το αίτημα του client πρέπει να μπορεί να εκφράσει μόνο την επιθυμητή κατηγορία πόρου και να ενεργοποιεί αποκλειστικά την προβλεπόμενη διαδικασία ανάθεσης.

Η αρχή αυτή περιορίζει σημαντικά την επιφάνεια πρόσβασης που εκτίθεται από την κεντρική υποδομή. Ο thin client δεν χρειάζεται να διαθέτει γενικά δικαιώματα διαχείρισης του hypervisor ούτε δυνατότητα αυθαίρετης εκτέλεσης εντολών. Η λεπτομερής σχεδίαση του διαχωρισμού δικαιωμάτων εξετάζεται στην Ενότητα 4.8.

Συνολικά, ο broker λειτουργεί ως μικρό και σαφώς οριοθετημένο επίπεδο ελέγχου. Αποκρύπτει από τον τελικό χρήστη τις λεπτομέρειες της υποδομής, συντονίζει την ασφαλή επιλογή διαθέσιμου πόρου και παρέχει στον thin client μόνο το αποτέλεσμα που απαιτείται για την έναρξη της συνεδρίας.

4.6 Σχεδίαση του Thin Client και Ροή Πρόσβασης Χρήστη

Ο thin client αποτελεί το μοναδικό τμήμα της αρχιτεκτονικής με το οποίο αλληλεπιδρά άμεσα ο τελικός χρήστης. Ο ρόλος του είναι σκόπιμα περιορισμένος: παρέχει μια απλή διεπαφή επιλογής λειτουργικού περιβάλλοντος, υποβάλλει το αντίστοιχο αίτημα στον broker και, μετά την επιτυχή ανάθεση, δημιουργεί τη συνεδρία απομακρυσμένης επιφάνειας εργασίας προς τον εικονικό σταθμό.

Η σχεδίαση αυτή αποσυνδέει το τερματικό από τη διαχείριση των ίδιων των εικονικών μηχανών. Ο thin client δεν χρειάζεται να γνωρίζει τη δομή των pools, τα αναγνωριστικά των VMs ή τον τρόπο με τον οποίο δημιουργούνται και επαναφέρονται οι εικονικοί σταθμοί. Από την πλευρά του χρήστη, το σύστημα εμφανίζεται ως μια απλή υπηρεσία επιλογής και πρόσβασης σε περιβάλλον Linux ή Windows.

4.6.1 Ροή πρόσβασης

Η αλληλεπίδραση του χρήστη σχεδιάζεται ως μια διαδοχική διαδικασία με όσο το δυνατόν λιγότερα απαιτούμενα βήματα. Μετά την εκκίνηση του thin client, ο χρήστης καλείται να επιλέξει το επιθυμητό λειτουργικό περιβάλλον. Η επιλογή μετατρέπεται σε αίτημα προς τον broker για το αντίστοιχο pool και ο thin client αναμένει το αποτέλεσμα της διαδικασίας ανάθεσης.

Η συνολική ροή μπορεί να αποδοθεί ως:

*Εκκίνηση thin client → Επιλογή Linux/Windows → Αίτημα στον broker → Ανάθεση VM →
Λήψη διεύθυνσης → Απομακρυσμένη συνεδρία*

Η πληροφορία που επιστρέφεται από τον broker περιορίζεται στη διεύθυνση του εικονικού σταθμού που έχει ανατεθεί. Πριν χρησιμοποιηθεί για την έναρξη της σύνδεσης, ο thin client ελέγχει ότι το αποτέλεσμα έχει την αναμενόμενη μορφή. Η λογική αυτή διατηρεί σαφή τη διαπαφή μεταξύ client και broker: επιτυχής ανάθεση αντιστοιχεί σε χρησιμοποιήσιμη πληροφορία σύνδεσης, ενώ μια αποτυχία δεν πρέπει να ερμηνεύεται ως έγκυρος προορισμός.

Μετά την επιτυχή ανάθεση, η σύνδεση απομακρυσμένης επιφάνειας εργασίας δημιουργείται απευθείας προς το επιλεγμένο VM. Από το σημείο αυτό και μετά, ο broker δεν συμμετέχει στην ενεργή συνεδρία και ο thin client λειτουργεί ως τερματικό του απομακρυσμένου λειτουργικού συστήματος.

Η προσέγγιση αυτή περιορίζει τις λειτουργίες που πρέπει να εκτελούνται τοπικά. Το τερματικό δεν φιλοξενεί το κύριο περιβάλλον εργασίας ούτε απαιτεί την εγκατάσταση των εφαρμογών που χρησιμοποιούνται στο εργαστήριο. Οι εφαρμογές και η κατάσταση του διαθέσιμου περιβάλλοντος βρίσκονται στην εικονική μηχανή, ενώ το τοπικό σύστημα αναλαμβάνει κυρίως την εκκίνηση της διαδικασίας πρόσβασης και την προβολή της απομακρυσμένης επιφάνειας εργασίας.

4.6.2 Ενοποιημένη πρόσβαση σε Windows και Linux

Μία από τις βασικές σχεδιαστικές επιλογές είναι η διατήρηση κοινής διαδικασίας πρόσβασης ανεξάρτητα από το λειτουργικό σύστημα που επιλέγει ο χρήστης. Η διαφορά μεταξύ Linux και Windows περιορίζεται στην επιλογή του αντίστοιχου pool και στις εσωτερικές ιδιαιτερότητες του guest. Η βασική ακολουθία από την πλευρά του thin client παραμένει ίδια.

Για τον σκοπό αυτό, τα δύο περιβάλλοντα παρέχουν συμβατή υπηρεσία απομακρυσμένης επιφάνειας εργασίας. Στα Windows χρησιμοποιείται η εγγενής υπηρεσία RDP, ενώ στα Linux VMs χρησιμοποιείται XRDP ώστε το περιβάλλον εργασίας να είναι επίσης προσβάσιμο μέσω RDP. Ο thin client μπορεί επομένως να χρησιμοποιεί κοινό RDP client και για τις δύο κατηγορίες εικονικών σταθμών.

Ο σχεδιασμός αυτός μειώνει τη διαφοροποίηση που πρέπει να γνωρίζει το τερματικό:

Linux επιλογή → Linux pool → XRDP

Windows επιλογή → Windows pool → RDP

και στις δύο περιπτώσεις όμως η διαδικασία από την πλευρά του χρήστη παραμένει:

Επιλογή περιβάλλοντος → Αυτόματη ανάθεση → Απομακρυσμένη επιφάνεια εργασίας

Η ενοποίηση αυτή είναι ιδιαίτερα σημαντική για τον ρόλο του thin client ως γενικού τερματικού πρόσβασης. Αντί να διατηρούνται διαφορετικοί clients ή διαφορετικές διαδικασίες σύνδεσης για κάθε λειτουργικό σύστημα, η διαφοροποίηση μεταφέρεται στην κεντρική υποδομή και στα αντίστοιχα pools.

4.6.3 Περιορισμός της τοπικής κατάστασης

Η αρχιτεκτονική αντιμετωπίζει τον thin client ως ελαφρύ και σε μεγάλο βαθμό αντικαταστάσιμο σημείο πρόσβασης. Δεν προβλέπεται η αποθήκευση της κατάστασης της εργασίας του χρήστη στο ίδιο το τερματικό ούτε η συσχέτισή του με συγκεκριμένο εικονικό σταθμό μεταξύ διαφορετικών συνεδριών.

Η προσέγγιση αυτή είναι συνεπής με το γενικό μοντέλο των pools. Όπως ο χρήστης δεν εξαρτάται από συγκεκριμένο VM, έτσι και η υπηρεσία πρόσβασης δεν πρέπει να εξαρτάται από σημαντική τοπική κατάσταση ενός συγκεκριμένου thin client. Το τερματικό χρειάζεται μόνο την απαραίτητη διαμόρφωση για την επικοινωνία με τον broker και την έναρξη της απομακρυσμένης συνεδρίας.

Η συγκεκριμένη σχεδίαση δεν εισάγει μηχανισμό προσωπικών λογαριασμών ή διατήρησης εξατομικευμένης κατάστασης χρήστη μεταξύ διαφορετικών VDI συνεδριών. Το σύστημα επικεντρώνεται στην κεντρική διάθεση προκαθορισμένων εργαστηριακών περιβαλλόντων και στην επαναφορά των εικονικών σταθμών σε γνωστή κατάσταση, αντί στην παροχή μόνιμου προσωπικού desktop για κάθε χρήστη.

4.7 Σχεδίαση Διαχείρισης Κύκλου Ζωής Εικονικών Μηχανών

Η λειτουργία του VDI περιβάλλοντος δεν περιορίζεται στην ανάθεση διαθέσιμων εικονικών σταθμών. Για να μπορούν τα pools να διατηρούνται σε προβλέψιμη και επαναχρησιμοποιήσιμη κατάσταση, απαιτείται ένα ξεχωριστό επίπεδο διαχείρισης του κύκλου ζωής των μελών τους. Η σχεδίαση του επιπέδου αυτού βασίζεται σε τέσσερις κύριες λειτουργίες: τη δημιουργία και αναπλήρωση των pools, την επαναφορά χρησιμοποιημένων σταθμών σε γνωστή καθαρή κατάσταση, την προστατευμένη απόσυρση διαχειριζόμενων εικονικών μηχανών και την παρακολούθηση της κατάστασης του περιβάλλοντος. Οι λειτουργίες αυτές χρησιμοποιούν κοινές πληροφορίες σχετικά με τα πρότυπα, τα pools, τα επιτρεπόμενα εύρη αναγνωριστικών και την κατάσταση ετοιμότητας των εικονικών μηχανών.

4.7.1 Δημιουργία και αναπλήρωση των pools

Η δημιουργία εικονικών σταθμών βασίζεται στην έννοια της επιθυμητής χωρητικότητας. Για κάθε λειτουργικό περιβάλλον καθορίζεται ένας επιθυμητός αριθμός διαχειριζόμενων μελών και η διαδικασία provisioning συγκρίνει την κατάσταση αυτή με τα μέλη που είναι ήδη καταχωρισμένα και εξακολουθούν να υπάρχουν στην υποδομή.

Όταν το πραγματικό μέγεθος του pool υπολείπεται του επιθυμητού, δημιουργούνται νέα linked clones από το αντίστοιχο πρότυπο. Κάθε νέο μέλος ακολουθεί τη διαδικασία προετοιμασίας που αντιστοιχεί στο λειτουργικό του σύστημα και αποκτά τα απαιτούμενα χαρακτηριστικά

ετοιμότητας και το καθαρό σημείο επαναφοράς. Μόνο μετά την ολοκλήρωση των σταδίων αυτών καταχωρίζεται στο pool και καθίσταται ορατό στον broker.

Η διαδικασία μπορεί να αποδοθεί αφαιρετικά ως:

Έλεγχος pool → Υπολογισμός ελλείμματος → Δημιουργία clone → Προετοιμασία → Κατάσταση ετοιμότητας → Καταχώριση

Η ίδια λειτουργία χρησιμοποιείται τόσο για την αρχική δημιουργία ενός pool όσο και για τη μεταγενέστερη αναπλήρωσή του. Επομένως, η διαχείριση δεν βασίζεται στην υπόθεση ότι συγκεκριμένα VMIDs πρέπει να υπάρχουν μόνιμα, αλλά στην επιδίωξη διατήρησης της καθορισμένης χωρητικότητας με έγκυρα μέλη.

Η προσέγγιση αυτή προσδίδει στη λειτουργία provisioning ιδιότητα επαναληψιμότητας. Όταν το pool βρίσκεται ήδη στην επιθυμητή κατάσταση, η εκ νέου εκτέλεση της διαδικασίας δεν απαιτεί τη δημιουργία πρόσθετων σταθμών. Αντίθετα, όταν ένα ή περισσότερα μέλη έχουν αφαιρεθεί, το έλλειμμα μπορεί να εντοπιστεί και να καλυφθεί με νέα clones.

4.7.2 Ασφαλής επαναφορά σε καθαρή κατάσταση

Μετά τη χρήση ενός εικονικού σταθμού, οι αλλαγές που έχουν πραγματοποιηθεί κατά τη συνεδρία δεν πρέπει να μεταφέρονται ανεξέλεγκτα στον επόμενο χρήστη. Για τον λόγο αυτό, η αρχιτεκτονική προβλέπει μηχανισμό επαναφοράς των διαχειριζόμενων VMs στο προκαθορισμένο καθαρό σημείο κατάστασης.

Η επαναφορά αποτελεί δυνητικά παρεμβατική λειτουργία και επομένως δεν πρέπει να πραγματοποιείται αποκλειστικά με βάση την ύπαρξη ενός VM μέσα στο pool. Πριν από την εκτέλεσή της απαιτείται να επιβεβαιώνεται ότι το VM εξακολουθεί να αποτελεί έγκυρο διαχειριζόμενο μέλος, διαθέτει τα αναμενόμενα χαρακτηριστικά ετοιμότητας και έχει το προβλεπόμενο σημείο επαναφοράς.

Ιδιαίτερη σημασία έχει η προστασία των ενεργών συνεδριών. Ένα VM που χρησιμοποιείται εκείνη τη στιγμή από τελικό χρήστη δεν πρέπει να επανέρχεται στην καθαρή κατάσταση. Για τον λόγο αυτό, η διαδικασία επαναφοράς λαμβάνει υπόψη όχι μόνο την κατάσταση λειτουργίας της εικονικής μηχανής αλλά και την ύπαρξη ενεργής απομακρυσμένης συνεδρίας.

Για σταθμούς που βρίσκονται ήδη σε σταματημένη κατάσταση, η επαναφορά μπορεί να πραγματοποιηθεί χωρίς να απαιτείται έλεγχος ενεργής συνεδρίας. Για ενεργούς σταθμούς, αντίθετα, απαιτείται πρώτα να διαπιστωθεί ότι δεν εξυπηρετούν χρήστη. Η σχεδίαση προβλέπει επίσης ένα ελάχιστο χρονικό περιθώριο μετά την εκκίνηση, ώστε ένας σταθμός που μόλις ανατέθηκε ή βρίσκεται ακόμη στη διαδικασία εκκίνησης να μην επιλεγεί πρόωρα για επαναφορά. Σε υψηλό επίπεδο, η απόφαση επαναφοράς ακολουθεί τη λογική:

Έγκυρο VDI μέλος → Έλεγχος κατάστασης → Προστασία ενεργής χρήσης → Αποκλειστική πρόσβαση → Rollback

Η ακριβής μέθοδος ανίχνευσης ενεργών συνεδριών διαφέρει μεταξύ Linux και Windows, χωρίς όμως να μεταβάλλεται η βασική αρχιτεκτονική αρχή: η απουσία ενεργού χρήστη αποτελεί προϋπόθεση για την επαναφορά ενός λειτουργούντος σταθμού.

4.7.3 Καταστροφή και αναπλήρωση διαχειριζόμενων μελών

Η καταστροφή μιας εικονικής μηχανής αποτελεί ξεχωριστή διοικητική λειτουργία και δεν χρησιμοποιείται ως ο συνήθης μηχανισμός καθαρισμού μεταξύ συνεδριών. Για την κανονική επαναχρησιμοποίηση ενός σταθμού προτιμάται η επαναφορά στο καθαρό snapshot, ενώ η καταστροφή προορίζεται για την ελεγχόμενη απόσυρση ενός διαχειριζόμενου μέλους ή ενός ολόκληρου pool.

Επειδή πρόκειται για μη αναστρέψιμη λειτουργία για το συγκεκριμένο clone, η προστασία δεν βασίζεται απλώς στο VMID που παρέχεται κατά την κλήση. Πριν επιτραπεί η καταστροφή, ελέγχεται ότι η εικονική μηχανή βρίσκεται στο επιτρεπόμενο εύρος αναγνωριστικών για το συγκεκριμένο λειτουργικό σύστημα, είναι καταχωρισμένη στο αντίστοιχο pool και διαθέτει τα αναμενόμενα lifecycle χαρακτηριστικά. Τα πρότυπα και τα συστήματα από τα οποία προέρχονται τα templates δεν αποτελούν επιτρεπόμενους στόχους της διαδικασίας.

Η ίδια λειτουργία μπορεί να εφαρμοστεί είτε σε συγκεκριμένο διαχειριζόμενο μέλος είτε σε όλα τα καταχωρισμένα μέλη ενός επιλεγμένου pool. Στη δεύτερη περίπτωση, κάθε VM εξακολουθεί να αντιμετωπίζεται ξεχωριστά και να υπόκειται στους ίδιους ελέγχους προστασίας πριν από την αφαίρεσή του. Η λειτουργία σε επίπεδο pool επομένως δεν παρακάμπτει τους ελέγχους της μεμονωμένης καταστροφής, αλλά εφαρμόζει την ίδια προστατευμένη διαδικασία διαδοχικά στα μέλη του pool.

Η καταχώριση ενός VM αφαιρείται από το pool μόνο στο πλαίσιο της επιτυχούς διαδικασίας απόσυρσής του. Μετά την αφαίρεση ενός ή περισσότερων μελών, η λειτουργία provisioning μπορεί να διαπιστώσει ότι η πραγματική χωρητικότητα βρίσκεται κάτω από τον καθορισμένο στόχο και να δημιουργήσει νέα μέλη από το αντίστοιχο template.

Προκύπτει έτσι ένας απλός κύκλος αντικατάστασης:

Pool → Απόσυρση μέλους → Έλλειμμα χωρητικότητας → Provisioning → Νέο έτοιμο μέλος

Η δυνατότητα αυτή ενισχύει την αντιμετώπιση των εικονικών σταθμών ως αντικαταστάσιμων μελών ενός διαχειριζόμενου συνόλου και όχι ως συστημάτων των οποίων η ατομική ταυτότητα πρέπει να διατηρείται μόνιμα.

4.7.4 Κοινός συγχρονισμός λειτουργιών

Οι λειτουργίες ανάθεσης, επαναφοράς και καταστροφής μπορούν όλες να μεταβάλουν ή να εξαρτώνται από την κατάσταση του ίδιου VM. Για τον λόγο αυτό δεν σχεδιάζονται ως ανεξάρτητες διαδικασίες που βασίζονται αποκλειστικά σε στιγμιαίους ελέγχους κατάστασης.

Η αρχιτεκτονική χρησιμοποιεί κοινό μηχανισμό αποκλεισμού ανά VM, τον οποίο οφείλουν να σέβονται όλες οι συνεργαζόμενες λειτουργίες. Μετά την απόκτηση του αντίστοιχου lock, η κρίσιμη κατάσταση του VM επανελέγχεται πριν πραγματοποιηθεί η ενέργεια.

Η κοινή αυτή πολιτική επιτρέπει την αποφυγή καταστάσεων όπως:

- ένα VM να ανατίθεται σε χρήστη ενώ ταυτόχρονα ξεκινά η επαναφορά του,
- ένα VM να καταστρέφεται ενώ βρίσκεται σε διαδικασία ανάθεσης,
- δύο διαφορετικές λειτουργίες διαχείρισης να μεταβάλλουν ταυτόχρονα το ίδιο μέλος.

Το lock επομένως αποτελεί κοινό μηχανισμό συντονισμού μεταξύ του broker και του lifecycle layer και όχι ιδιωτική λεπτομέρεια μιας συγκεκριμένης λειτουργίας.

4.7.5 Παρακολούθηση της κατάστασης

Εκτός από τις λειτουργίες που μεταβάλλουν το VDI περιβάλλον, απαιτείται και δυνατότητα παρατήρησης της τρέχουσας κατάστασής του χωρίς την πρόκληση αλλαγών. Για τον σκοπό αυτό προβλέπεται ξεχωριστή λειτουργία παρακολούθησης.

Η προβολή κατάστασης συγκεντρώνει τις πληροφορίες που χρειάζεται ο διαχειριστής για να αξιολογήσει την κατάσταση των pools και των μελών τους, όπως τον αριθμό καταχωρισμένων σταθμών σε σχέση με την επιθυμητή χωρητικότητα, την κατάσταση λειτουργίας των VMs και τις βασικές ενδείξεις που σχετίζονται με την ετοιμότητα και τη χρήση τους.

Η λειτουργία αυτή είναι σκόπιμα μόνο για ανάγνωση. Δεν πραγματοποιεί επαναφορά, δημιουργία ή καταστροφή εικονικών μηχανών, αλλά παρέχει μια ενιαία εικόνα πάνω στην οποία μπορούν να βασιστούν οι επόμενες διαχειριστικές ενέργειες.

Με τον συνδυασμό provisioning, reset, destruction και status, το lifecycle layer συμπληρώνει τον μηχανισμό ανάθεσης και επιτρέπει τη διαχείριση των εικονικών σταθμών ως ενός συνεκτικού συνόλου πόρων με καθορισμένη κατάσταση, χωρητικότητα και κανόνες προστασίας.

4.8 Ασφάλεια και Διαχωρισμός Δικαιωμάτων

Η ασφάλεια της προτεινόμενης αρχιτεκτονικής βασίζεται κυρίως στον περιορισμό των δυνατοτήτων που παρέχονται σε κάθε επιμέρους ρόλο. Ο thin client, ο μηχανισμός ανάθεσης και οι διοικητικές λειτουργίες του κύκλου ζωής δεν απαιτείται να διαθέτουν το ίδιο επίπεδο πρόσβασης στην υποδομή. Για τον λόγο αυτό, η σχεδίαση επιδιώκει να διαχωρίσει τη διαδρομή πρόσβασης του τελικού χρήστη από τις λειτουργίες που μπορούν να μεταβάλουν την κατάσταση του hypervisor και των εικονικών μηχανών.

Το μοντέλο αυτό δεν επιχειρεί να αντικαταστήσει ένα ολοκληρωμένο σύστημα κεντρικής ταυτοποίησης, εξουσιοδότησης και ελέγχου πρόσβασης. Αντίθετα, εφαρμόζει περιορισμούς κατάλληλους για το συγκεκριμένο πειραματικό VDI περιβάλλον, με στόχο να αποφεύγεται η παροχή γενικών διαχειριστικών δυνατοτήτων σε στοιχεία που χρειάζονται μόνο ένα μικρό υποσύνολο λειτουργιών.

4.8.1 Περιορισμένη διαδρομή πρόσβασης

Ο thin client χρειάζεται να μπορεί να ζητήσει έναν διαθέσιμο σταθμό Linux ή Windows, αλλά δεν υπάρχει λειτουργική ανάγκη να αποκτά γενική απομακρυσμένη πρόσβαση στο σύστημα που φιλοξενεί τον broker και τις εικονικές μηχανές.

Για τον λόγο αυτό, η διαδρομή πρόσβασης σχεδιάζεται έτσι ώστε η αυθεντικοποίηση του thin client να οδηγεί σε προκαθορισμένο σημείο εισόδου. Το σημείο αυτό αποδέχεται μόνο την πληροφορία που απαιτείται για την επιλογή του αντίστοιχου pool και ενεργοποιεί την προβλεπόμενη διαδικασία ανάθεσης. Ο client δεν χρειάζεται να αποκτά διαδραστικό κέλυφος ούτε να εκτελεί αυθαίρετες εντολές στο κεντρικό σύστημα.

Η λογική της διαδρομής μπορεί να αποδοθεί ως:

Thin client → *Περιορισμένο σημείο εισόδου* → *Broker* → *Επιτρεπόμενη λειτουργία ανάθεσης*

Η προσέγγιση αυτή διαχωρίζει την κατοχή των διαπιστευτηρίων που απαιτούνται για την υποβολή ενός αιτήματος από τη δυνατότητα γενικής διαχείρισης της υποδομής. Ακόμη και αν το

τερματικό μπορεί να ενεργοποιήσει τη διαδικασία ανάθεσης, η συγκεκριμένη δυνατότητα δεν πρέπει από μόνη της να συνεπάγεται πρόσβαση σε άλλες διοικητικές λειτουργίες του host.

4.8.2 Διαχωρισμός ρόλων και δικαιωμάτων

Στο εσωτερικό του κεντρικού συστήματος διακρίνονται διαφορετικοί ρόλοι εκτέλεσης. Το σημείο εισόδου που χρησιμοποιείται από τον thin client δεν εκτελεί απευθείας όλες τις απαιτούμενες λειτουργίες διαχείρισης. Αντίθετα, η αίτηση μεταβιβάζεται στον κατάλληλο μηχανισμό, ο οποίος διαθέτει μόνο τα δικαιώματα που απαιτούνται για την προβλεπόμενη λειτουργία του. Η σχεδίαση ακολουθεί επομένως μια αλυσίδα περιορισμένης μεταβίβασης:

Client identity → *Restricted entry point* → *Broker identity* → *Ελεγχόμενες λειτουργίες VM*

Η κλιμάκωση δικαιωμάτων περιορίζεται στις συγκεκριμένες ενέργειες που απαιτούνται για την ανάθεση και τη διαχείριση των VDI πόρων. Δεν αποτελεί γενική εξουσιοδότηση εκτέλεσης οποιασδήποτε διαχειριστικής εντολής.

Η αρχή αυτή είναι ιδιαίτερα σημαντική επειδή ο broker λειτουργεί στο ίδιο κεντρικό περιβάλλον με τον hypervisor. Η τοποθέτησή του εκεί απλοποιεί την πρόσβαση στις λειτουργίες διαχείρισης των VMs, αλλά ταυτόχρονα καθιστά αναγκαίο τον σαφή περιορισμό της διαδρομής μέσω της οποίας ένα αίτημα από τον thin client μπορεί να οδηγήσει σε προνομιούχα ενέργεια.

4.8.3 Προστασία διαχειριστικών λειτουργιών

Οι λειτουργίες επαναφοράς και καταστροφής μπορούν να μεταβάλουν σημαντικά ή μη αναστρέψιμα την κατάσταση μιας εικονικής μηχανής. Η προστασία τους επομένως βασίζεται σε περισσότερους από έναν ελέγχους και όχι μόνο στην παροχή ενός VMID.

Για τις διαχειριζόμενες λειτουργίες χρησιμοποιούνται πληροφορίες όπως:

- η καταχώριση του VM στο αναμενόμενο pool,
- η ένταξη του αναγνωριστικού του στο επιτρεπόμενο εύρος,
- η παρουσία των αναμενόμενων lifecycle χαρακτηριστικών,
- η ύπαρξη του προβλεπόμενου σημείου επαναφοράς όπου αυτό απαιτείται,
- η κατάσταση λειτουργίας του VM και, ειδικά για την αυτόματη επαναφορά, η ύπαρξη ενεργής συνεδρίας χρήστη,
- η δυνατότητα απόκτησης αποκλειστικού lock πριν από μια μεταβολή.

Οι έλεγχοι αυτοί λειτουργούν συνδυαστικά. Η ύπαρξη ενός VM στον hypervisor δεν αρκεί για να θεωρηθεί έγκυρος στόχος μιας VDI διοικητικής λειτουργίας. Αντίστοιχα, η γνώση ενός αριθμού VMID δεν πρέπει να επιτρέπει την καταστροφή ενός template, ενός συστήματος προετοιμασίας ή μιας άλλης εικονικής μηχανής που δεν ανήκει στο διαχειριζόμενο pool.

Η καταστροφή προστατεύεται μέσω της σκόπιμης χρήσης της αντίστοιχης διοικητικής λειτουργίας και των παραπάνω ελέγχων εγκυρότητας. Δεν βασίζεται σε διαδραστική επιβεβαίωση

από τον διαχειριστή. Η ίδια πολιτική εφαρμόζεται είτε η λειτουργία αφορά ένα συγκεκριμένο μέλος είτε την ελεγχόμενη αφαίρεση όλων των μελών ενός pool.

Ο κοινός μηχανισμός κλειδώματος προσθέτει ένα διαφορετικό επίπεδο προστασίας. Οι στατικοί έλεγχοι καθορίζουν αν μια ενέργεια επιτρέπεται να εφαρμοστεί στο συγκεκριμένο αντικείμενο, ενώ το lock αποτρέπει τη σύγκρουσή της με άλλη συνεργαζόμενη λειτουργία που βρίσκεται ήδη σε εξέλιξη για το ίδιο VM.

4.8.4 Διαχωρισμός επιπέδου ελέγχου και συνεδρίας χρήστη

Ένα επιπλέον όριο της αρχιτεκτονικής προκύπτει από τον διαχωρισμό μεταξύ της διαδικασίας ανάθεσης και της τελικής συνεδρίας απομακρυσμένης επιφάνειας εργασίας.

Ο broker συμμετέχει στη ροή ελέγχου μόνο μέχρι να επιλεγεί ο κατάλληλος σταθμός και να επιστραφεί η διεύθυνσή του. Η γραφική συνεδρία πραγματοποιείται στη συνέχεια απευθείας μεταξύ thin client και guest. Επομένως, τα δικαιώματα που απαιτούνται για την ανάθεση ενός VM παραμένουν διαχωρισμένα από την πρόσβαση του χρήστη στο λειτουργικό σύστημα του ίδιου του VM.

Η διάκριση αυτή δεν αποτελεί από μόνη της μηχανισμό αυθεντικοποίησης του τελικού χρήστη, αλλά περιορίζει τον ρόλο του broker και διατηρεί σαφέστερα τα όρια μεταξύ του management plane και της πραγματικής συνεδρίας εργασίας.

4.8.5 Περιορισμοί του μοντέλου ασφάλειας

Παρά τους παραπάνω περιορισμούς, η σχεδίαση δεν πρέπει να θεωρηθεί ολοκληρωμένο μοντέλο ασφάλειας παραγωγικού VDI περιβάλλοντος. Το σύστημα αναπτύχθηκε με προτεραιότητα την αυτοματοποίηση της διάθεσης και διαχείρισης κοινών εργαστηριακών σταθμών και όχι την υλοποίηση εξατομικευμένης διαχείρισης ταυτότητας.

Δεν υλοποιείται κεντρικός μηχανισμός αντιστοίχισης κάθε συνεδρίας σε μοναδικό λογαριασμό χρήστη ούτε προσωπικό persistent desktop. Η πρόσβαση στα guest περιβάλλοντα βασίζεται σε προκαθορισμένους εργαστηριακούς λογαριασμούς. Συνεπώς, το σύστημα δεν παρέχει από μόνο του δυνατότητες όπως λεπτομερή λογιστικό έλεγχο ενεργειών ανά φυσικό χρήστη, εξατομικευμένες πολιτικές πρόσβασης ή μόνιμη προσωπική κατάσταση μεταξύ συνεδριών.

Επιπλέον, η εφαρμογή της αρχής των ελάχιστων δικαιωμάτων δεν είναι απόλυτη σε όλα τα επίπεδα της πειραματικής υλοποίησης. Ειδικότερα, ο λογαριασμός που χρησιμοποιείται στο Linux guest διαθέτει ευρύτερα τοπικά δικαιώματα από όσα θα ήταν επιθυμητά σε μια αυστηρότερα περιορισμένη εργαστηριακή εγκατάσταση. Η επιλογή αυτή δεν αποτελεί απαίτηση της αρχιτεκτονικής και μπορεί να περιοριστεί σε μελλοντική ανάπτυξη χωρίς να μεταβληθεί η βασική δομή του συστήματος.

Σε περιβάλλον παραγωγής θα ήταν επίσης σκόπιμη η ενσωμάτωση ισχυρότερων μηχανισμών ταυτοποίησης, διαχείρισης διαπιστευτηρίων, καταγραφής ενεργειών και κεντρικής πολιτικής πρόσβασης. Οι δυνατότητες αυτές θεωρούνται επεκτάσεις του παρόντος μοντέλου και όχι λειτουργίες που υλοποιήθηκαν στο πλαίσιο της εργασίας.

4.9 Όρια Σχεδίασης και Δυνατότητες Επέκτασης

Η προτεινόμενη αρχιτεκτονική σχεδιάστηκε με στόχο την αυτοματοποίηση της διάθεσης και της διαχείρισης εικονικών σταθμών εργασίας σε περιβάλλον εκπαιδευτικού εργαστηρίου. Το πεδίο εφαρμογής της εργασίας επικεντρώνεται στη δημιουργία μιας λειτουργικής και σχετικά απλής υποδομής, η οποία συνδυάζει thin clients, κεντρική ανάθεση πόρων, διακριτά Windows και Linux pools και αυτοματοποιημένες λειτουργίες διαχείρισης του κύκλου ζωής των εικονικών μηχανών.

Η σχεδίαση αυτή συνεπάγεται συγκεκριμένα όρια, τα οποία πρέπει να διαχωρίζονται από ενδεχόμενες δυνατότητες μελλοντικής επέκτασης.

4.9.1 Όρια της παρούσας σχεδίασης

Η κεντρική υποδομή βασίζεται σε έναν Proxmox VE host, ο οποίος φιλοξενεί τόσο τα πρότυπα και τις εικονικές μηχανές όσο και τους βασικούς μηχανισμούς ανάθεσης και διαχείρισης. Κατά συνέπεια, η αρχιτεκτονική δεν παρέχει υψηλή διαθεσιμότητα σε επίπεδο hypervisor. Μια αστοχία ή διακοπή λειτουργίας του συγκεκριμένου host επηρεάζει τη συνολική διαθεσιμότητα της υπηρεσίας VDI.

Αντίστοιχα, δεν υλοποιείται κατανεμημένη διαχείριση των pools μεταξύ πολλαπλών κόμβων. Η επιλογή και ο κύκλος ζωής των εικονικών σταθμών εξετάζονται στο πλαίσιο ενός κεντρικού περιβάλλοντος και δεν περιλαμβάνονται μηχανισμοί μεταφοράς φορτίου, επιλογής hypervisor ή συντονισμού κοινών pools μεταξύ διαφορετικών φυσικών εξυπηρετητών.

Η χωρητικότητα κάθε pool καθορίζεται μέσω προκαθορισμένης επιθυμητής τιμής. Η διαδικασία provisioning μπορεί να αποκαταστήσει αυτή τη χωρητικότητα όταν λείπουν μέλη, αλλά δεν υλοποιείται δυναμική προσαρμογή του επιθυμητού μεγέθους με βάση τον αριθμό των χρηστών, το τρέχον φορτίο ή άλλες μετρικές. Επομένως, η αναπλήρωση της καθορισμένης χωρητικότητας δεν πρέπει να συγχέεται με μηχανισμό δυναμικού autoscaling.

Επιπλέον, η αρχιτεκτονική δεν υλοποιεί εξατομικευμένο persistent VDI. Τα μέλη των pools αντιμετωπίζονται ως επαναχρησιμοποιήσιμοι σταθμοί κοινής χρήσης και μπορούν να επανέρχονται στην προκαθορισμένη καθαρή τους κατάσταση. Δεν υπάρχει μόνιμη αντιστοίχιση ενός συγκεκριμένου VM σε συγκεκριμένο χρήστη ούτε μηχανισμός διατήρησης προσωπικού desktop μεταξύ διαφορετικών συνεδριών.

Η πρόσβαση επίσης δεν βασίζεται σε ολοκληρωμένο σύστημα κεντρικής διαχείρισης ταυτότητας. Η σχεδίαση επικεντρώνεται στην ασφαλή ενεργοποίηση της διαδικασίας ανάθεσης και στη σύνδεση σε προκαθορισμένα εργαστηριακά περιβάλλοντα, χωρίς να εισάγει μηχανισμούς εξατομικευμένης αυθεντικοποίησης, πολιτικών πρόσβασης ανά χρήστη ή κεντρικής διαχείρισης προσωπικών προφίλ.

Τέλος, η αρχιτεκτονική και η υλοποίησή της αξιολογούνται στο πλαίσιο ελεγχόμενου πειραματικού περιβάλλοντος. Η εργασία δεν περιλαμβάνει ανάπτυξη της λύσης σε πλήρη παραγωγική κλίμακα εκπαιδευτικού εργαστηρίου ούτε συστηματική αξιολόγηση με μεγάλο αριθμό ταυτόχρονων πραγματικών χρηστών. Για τον λόγο αυτό, συμπεράσματα σχετικά με τη λειτουργία σε μεγαλύτερη κλίμακα πρέπει να αντιμετωπίζονται ως αντικείμενο περαιτέρω αξιολόγησης και όχι ως ήδη επαληθευμένο χαρακτηριστικό της λύσης.

4.9.2 Δυνατότητες επέκτασης

Η διάκριση μεταξύ thin client, broker, pools και lifecycle management επιτρέπει την επέκταση επιμέρους τμημάτων χωρίς να απαιτείται απαραίτητα πλήρης ανασχεδίαση της βασικής ροής πρόσβασης.

Μια πιθανή επέκταση αφορά την υποστήριξη περισσότερων κόμβων εικονικοποίησης. Σε μια τέτοια περίπτωση, ο μηχανισμός ανάθεσης θα μπορούσε να λαμβάνει υπόψη όχι μόνο τη διαθεσιμότητα ενός VM αλλά και τον κόμβο στον οποίο αυτό φιλοξενείται, ενώ οι λειτουργίες provisioning και lifecycle management θα έπρεπε να επεκταθούν ώστε να συντονίζουν πόρους σε επίπεδο cluster. Η κατεύθυνση αυτή θα μπορούσε επίσης να συνδυαστεί με μηχανισμούς υψηλής διαθεσιμότητας.

Μια δεύτερη κατεύθυνση είναι η δυναμική προσαρμογή της χωρητικότητας. Αντί ο επιθυμητός αριθμός μελών κάθε pool να παραμένει σταθερά καθορισμένος, θα μπορούσε να υπολογίζεται με βάση τη ζήτηση, το ωράριο λειτουργίας του εργαστηρίου ή τη χρήση των διαθέσιμων πόρων. Ένας τέτοιος μηχανισμός θα μετέτρεπε την υπάρχουσα λογική αναπλήρωσης σε ευρύτερο σύστημα autoscaling.

Η διαχείριση ταυτότητας αποτελεί επίσης σημαντική δυνατότητα επέκτασης. Η ενσωμάτωση κεντρικής υπηρεσίας καταλόγου ή άλλου μηχανισμού αυθεντικοποίησης θα μπορούσε να επιτρέψει μοναδικούς λογαριασμούς χρηστών, λεπτομερέστερο έλεγχο πρόσβασης και καλύτερη συσχέτιση των συνεδριών με συγκεκριμένους χρήστες. Ανάλογα με τις απαιτήσεις, η επέκταση αυτή θα μπορούσε να συνδυαστεί με απομακρυσμένα προφίλ ή άλλες τεχνικές διατήρησης προσωπικών δεδομένων χωρίς να απαιτείται μόνιμη αντιστοίχιση χρήστη και VM.

Μια επιπλέον δυνατότητα αφορά την ανάπτυξη κεντρικής διεπαφής διαχείρισης. Οι λειτουργίες provisioning, reset, destruction και status θα μπορούσαν να εκτεθούν μέσω κατάλληλου διαχειριστικού API ή γραφικής διεπαφής, ώστε ο διαχειριστής να μην χρειάζεται να αλληλεπιδρά άμεσα με τις επιμέρους λειτουργίες του host. Μια τέτοια διεπαφή θα έπρεπε να διατηρεί τους ίδιους ελέγχους εγκυρότητας, συγχρονισμού και διαχωρισμού δικαιωμάτων που εφαρμόζονται στο υφιστάμενο lifecycle layer.

Αντίστοιχα, η πρόσβαση των τελικών χρηστών θα μπορούσε να επεκταθεί πέρα από το μοντέλο του ειδικού thin client, για παράδειγμα μέσω κατάλληλης web-based πύλης απομακρυσμένης πρόσβασης. Μια τέτοια προσέγγιση θα άλλαζε το επίπεδο παρουσίας της υπηρεσίας, χωρίς να απαιτεί απαραίτητα την κατάργηση της βασικής λογικής των pools και του broker.

Τέλος, η μετάβαση σε πραγματική εργαστηριακή εγκατάσταση θα επέτρεπε τη συστηματική μελέτη χαρακτηριστικών που δεν μπορούν να αξιολογηθούν πλήρως στο παρόν πειραματικό περιβάλλον. Σε αυτά περιλαμβάνονται η συμπεριφορά με μεγαλύτερο αριθμό ταυτόχρονων συνεδριών, η κατανάλωση υπολογιστικών και αποθηκευτικών πόρων, η δικτυακή επιβάρυνση, η εμπειρία χρήσης σε αντιπροσωπευτικό φυσικό εξοπλισμό thin client και η συμπεριφορά του συστήματος σε μεγαλύτερη χρονική διάρκεια.

Οι παραπάνω κατευθύνσεις δεν αποτελούν λειτουργίες της παρούσας υλοποίησης. Δείχνουν, ωστόσο, ότι ο διαχωρισμός των βασικών ρόλων της αρχιτεκτονικής παρέχει ένα σημείο εκκίνησης πάνω στο οποίο μπορούν να προστεθούν πιο σύνθετες δυνατότητες, εφόσον αυξηθούν οι απαιτήσεις κλίμακας, διαθεσιμότητας, ασφάλειας ή διαχείρισης.

4.10 Σύνοψη Κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε η σχεδίαση και η συνολική αρχιτεκτονική του προτεινόμενου συστήματος VDI. Η προσέγγιση βασίζεται στον διαχωρισμό της πρόσβασης του χρήστη, της ανάθεσης υπολογιστικών πόρων και της διαχείρισης του κύκλου ζωής των εικονικών σταθμών, ώστε οι επιμέρους λειτουργίες να παραμένουν σαφώς οριοθετημένες και να μπορούν να αυτοματοποιηθούν ανεξάρτητα.

Οι εικονικοί σταθμοί οργανώνονται σε διακριτά Linux και Windows pools και δημιουργούνται από προκαθορισμένα πρότυπα ως αντικαταστάσιμα μέλη ενός κεντρικά διαχειριζόμενου συνόλου πόρων. Ο broker μεσολαβεί μεταξύ των thin clients και των pools, αναλαμβάνοντας την επιλογή και ασφαλή δέσμευση ενός διαθέσιμου σταθμού, την εκκίνησή του και την επιστροφή της δικτυακής πληροφορίας που απαιτείται για τη σύνδεση. Η πραγματική συνεδρία απομακρυσμένης επιφάνειας εργασίας πραγματοποιείται στη συνέχεια απευθείας μεταξύ thin client και εικονικής μηχανής.

Παράλληλα, το επίπεδο διαχείρισης του κύκλου ζωής υποστηρίζει τη δημιουργία και αναπλήρωση των pools, την ελεγχόμενη επαναφορά των εικονικών σταθμών σε γνωστή καθαρή κατάσταση, την προστατευμένη απόσυρση μεμονωμένων μελών ή ολόκληρου pool και την παρακολούθηση της κατάστασης του περιβάλλοντος. Ο κοινός μηχανισμός συγχρονισμού ανά εικονική μηχανή συνδέει τις λειτουργίες αυτές με τον μηχανισμό ανάθεσης και περιορίζει τις συγκρούσεις μεταξύ ταυτόχρονων ενεργειών.

Η σχεδίαση συμπληρώνεται από τον περιορισμό της διαδρομής πρόσβασης από τον thin client και τον διαχωρισμό των δικαιωμάτων που απαιτούνται από τους διαφορετικούς ρόλους του συστήματος. Παράλληλα, αναγνωρίζονται τα όρια της παρούσας προσέγγισης, όπως η εξάρτηση από έναν κεντρικό κόμβο εικονικοποίησης, η απουσία μηχανισμών υψηλής διαθεσιμότητας και δυναμικού autoscaling, καθώς και η μη υλοποίηση εξατομικευμένης διαχείρισης ταυτότητας και persistent προσωπικών desktops.

Με βάση την αρχιτεκτονική αυτή, το επόμενο κεφάλαιο παρουσιάζει την πρακτική υλοποίηση του συστήματος. Ειδικότερα, περιγράφονται η διαμόρφωση της υποδομής και των προτύπων, η υλοποίηση των Windows και Linux pools, ο broker και η διαδρομή πρόσβασης από τον thin client, καθώς και οι μηχανισμοί αυτοματοποίησης που χρησιμοποιήθηκαν για τη διαχείριση του κύκλου ζωής των εικονικών μηχανών.

Κεφάλαιο 5

Υλοποίηση του Συστήματος

5.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο παρουσιάστηκε ο σχεδιασμός και η αρχιτεκτονική του προτεινόμενου συστήματος VDI, με έμφαση στον διαχωρισμό των επιμέρους υποσυστημάτων, στις μεταξύ τους αλληλεπιδράσεις και στις βασικές σχεδιαστικές επιλογές. Στο παρόν κεφάλαιο παρουσιάζεται η πρακτική υλοποίηση της αρχιτεκτονικής αυτής στο πειραματικό περιβάλλον που αναπτύχθηκε στο πλαίσιο της εργασίας.

Η υλοποίηση βασίστηκε σε έναν εξυπηρετητή Proxmox VE, στον οποίο φιλοξενήθηκαν τα πρότυπα λειτουργικών συστημάτων και οι εικονικές μηχανές των VDI pools, καθώς και ο μηχανισμός ανάθεσης και διαχείρισης των διαθέσιμων πόρων. Παράλληλα, διαμορφώθηκε ένας ελαφρύς thin client, μέσω του οποίου ο χρήστης επιλέγει περιβάλλον Linux ή Windows και συνδέεται στην εικονική μηχανή που του ανατίθεται. Για την υποστήριξη της συνολικής λειτουργίας υλοποιήθηκαν επιπλέον εργαλεία για τη δημιουργία και διατήρηση των pools, την επαναφορά των εικονικών μηχανών σε προκαθορισμένη καθαρή κατάσταση, την ελεγχόμενη καταστροφή τους και την παρακολούθηση της κατάστασης του συστήματος.

Στις ενότητες που ακολουθούν περιγράφεται αρχικά το πειραματικό περιβάλλον και η βασική υποδομή πάνω στην οποία πραγματοποιήθηκε η υλοποίηση. Στη συνέχεια παρουσιάζεται η προετοιμασία των Windows και Linux golden images και η μετατροπή τους σε πρότυπα κατάλληλα για τη δημιουργία linked clones. Ακολουθεί η υλοποίηση του broker και του thin client, καθώς και των μηχανισμών διαχείρισης του κύκλου ζωής των εικονικών μηχανών. Τέλος, παρουσιάζεται η ολοκλήρωση των επιμέρους υποσυστημάτων και ορισμένα πρακτικά ζητήματα που προέκυψαν κατά την ανάπτυξη και επηρέασαν την τελική μορφή της υλοποίησης.

Το κεφάλαιο επικεντρώνεται στο τι διαμορφώθηκε και υλοποιήθηκε και στον τρόπο με τον οποίο πραγματοποιήθηκε η τεχνική ολοκλήρωση των επιμέρους μηχανισμών. Η συστηματική επαλήθευση της λειτουργίας τους, τα αποτελέσματα των δοκιμών και η αξιολόγηση της συνολικής λύσης παρουσιάζονται χωριστά στο επόμενο κεφάλαιο.

5.2 Πειραματικό Περιβάλλον και Υποδομή

Η υλοποίηση πραγματοποιήθηκε σε πειραματικό περιβάλλον βασισμένο στην πλατφόρμα Proxmox VE. Ο κεντρικός εξυπηρετητής είχε hostname IT-DHCP-35 και κατά την τελική

καταγραφή εκτελούσε Proxmox VE 8.4.0 με kernel 6.8.12-9-pve. Στον ίδιο εξυπηρετητή φιλοξενήθηκαν τα golden images, τα τελικά templates, οι εικονικές μηχανές των VDI pools και οι μηχανισμοί broker και lifecycle management.

Η βασική δικτυακή πρόσβαση του Proxmox host πραγματοποιήθηκε μέσω της γέφυρας `vmbr0`, η οποία ήταν συνδεδεμένη με τη φυσική διεπαφή `ens18`. Ο host χρησιμοποιούσε τη διεύθυνση `192.168.6.35/24`, με προεπιλεγμένη πύλη τη `192.168.6.1`. Στο ίδιο δίκτυο συνδέθηκαν οι εικονικές μηχανές που χρησιμοποιήθηκαν για την τελική VDI ροή, ώστε οι thin clients να μπορούν να προσπελάσουν απευθείας τις υπηρεσίες RDP ή XRDP των αντίστοιχων guests.

Παράλληλα, υπήρχε η εσωτερική γέφυρα `vmbr1`, με διεύθυνση `10.10.10.1/24`. Η συγκεκριμένη διεπαφή χρησιμοποιήθηκε στο πλαίσιο της πειραματικής υποδομής ως εσωτερικό δίκτυο, με διαμορφωμένη δυνατότητα προώθησης IPv4 και NAT. Η τελική VDI ροή που εξετάζεται στο παρόν κεφάλαιο βασίστηκε κυρίως στο δίκτυο της `vmbr0`.

Ο broker δεν υλοποιήθηκε ως ανεξάρτητη εικονική μηχανή. Τα σχετικά scripts και οι λογαριασμοί υπηρεσίας εγκαταστάθηκαν απευθείας στον Proxmox host. Για τον διαχωρισμό των αρμοδιοτήτων χρησιμοποιήθηκαν δύο διαφορετικοί λογαριασμοί: ο `lab-launcher`, ο οποίος λειτουργεί ως περιορισμένο σημείο εισόδου μέσω SSH για τα αιτήματα των thin clients, και ο `vdi-broker`, υπό τον οποίο εκτελείται ο allocator με τις συγκεκριμένες εξουσιοδοτήσεις που απαιτούνται για την αλληλεπίδραση με τις εικονικές μηχανές.

Η τελική υλοποίηση περιλάμβανε δύο ξεχωριστές κατηγορίες VDI περιβαλλόντων. Για το Linux χρησιμοποιήθηκε το template με VMID 110, ενώ για τα Windows το template με VMID 111. Για τα διαχειριζόμενα clones δεσμεύθηκαν οι περιοχές VMID 120--129 και 130--139 αντίστοιχα. Στην τελική πειραματική διαμόρφωση κάθε pool είχε επιθυμητό μέγεθος δύο εικονικών μηχανών.

Οι βασικές παράμετροι της υποδομής και του lifecycle management συγκεντρώθηκαν στο αρχείο `/etc/vdi-broker/lifecycle.conf`. Με αυτόν τον τρόπο οι ταυτότητες των templates, τα επιθυμητά μεγέθη των pools, οι δεσμευμένες περιοχές VMID, τα prefixes ονοματοδοσίας, τα αρχεία καταχώρισης των pools και οι παράμετροι της κατάστασης `rdp-ready` ορίστηκαν σε κοινό σημείο και χρησιμοποιήθηκαν από τα επιμέρους εργαλεία διαχείρισης.

Συνοπτικά, οι κύριες παράμετροι της τελικής διαμόρφωσης παρουσιάζονται στον Πίνακα 5.1.

Πίνακας 5.1: Βασικές παράμετροι του πειραματικού περιβάλλοντος

Παράμετρος	Τελική διαμόρφωση
Πλατφόρμα εικονικοποίησης	Proxmox VE 8.4.0
Proxmox host	IT-DHCP-35
Κύρια διεύθυνση host	192.168.6.35/24
Κύρια γέφυρα	vmbr0
Εσωτερική γέφυρα	vmbr1, 10.10.10.1/24
Linux template	VMID 110
Windows template	VMID 111
Linux managed range	VMID 120--129
Windows managed range	VMID 130--139
Linux pool target	2 VMs
Windows pool target	2 VMs
Lifecycle configuration	/etc/vdi-broker/lifecycle.conf

Η παραπάνω υποδομή αποτέλεσε τη βάση πάνω στην οποία πραγματοποιήθηκε η προετοιμασία των λειτουργικών συστημάτων, η δημιουργία των templates και των linked clones, καθώς και η εγκατάσταση των μηχανισμών broker, thin client και lifecycle management που παρουσιάζονται στις επόμενες ενότητες.

5.3 Προετοιμασία των Golden Images και Templates

Η δημιουργία των VDI pools βασίστηκε σε δύο προετοιμασμένα λειτουργικά περιβάλλοντα, ένα για Windows και ένα για Linux. Για κάθε λειτουργικό σύστημα διατηρήθηκε αρχικά μία μεταβλητή εικονική μηχανή ως golden image, στην οποία πραγματοποιήθηκε η απαραίτητη εγκατάσταση και διαμόρφωση του guest λειτουργικού συστήματος. Μετά την ολοκλήρωση της προετοιμασίας δημιουργήθηκε το αντίστοιχο Proxmox template, το οποίο χρησιμοποιήθηκε ως βάση για την παραγωγή των διαχειριζόμενων VDI clones.

5.3.1 Προετοιμασία του Windows Golden Image

Ως τελική πηγή προετοιμασίας των Windows χρησιμοποιήθηκε η εικονική μηχανή VMID 108, με όνομα WindowsGI. Το λειτουργικό σύστημα ήταν Windows 10, έκδοση 10.0.19045.6456. Στο guest εγκαταστάθηκαν και διαμορφώθηκαν τα απαραίτητα VirtIO/QEMU components, ενώ το QEMU Guest Agent εγκαταστάθηκε ως υπηρεσία με αυτόματη εκκίνηση. Με αυτόν τον τρόπο ο Proxmox host μπορούσε να επικοινωνεί με το guest και να ανακτά πληροφορίες που απαιτούνται από τους μηχανισμούς διαχείρισης.

Για την απομακρυσμένη πρόσβαση ενεργοποιήθηκε το Remote Desktop. Η υπηρεσία TermService ήταν ενεργή και οι αντίστοιχοι κανόνες του Windows Firewall για εισερχόμενες συνδέσεις RDP είχαν ενεργοποιηθεί. Ως κοινός λογαριασμός του πειραματικού VDI περιβάλλοντος χρησιμοποιήθηκε ο τοπικός χρήστης Studin, για τον οποίο διαμορφώθηκε επίσης αυτόματη σύνδεση στο λειτουργικό σύστημα.

Κατά την τελική προετοιμασία διαπιστώθηκε ότι ο κωδικός πρόσβασης του λογαριασμού Studin είχε ημερομηνία λήξης. Επειδή η λήξη του κωδικού θα μπορούσε να διακόψει την αυτοματοποιημένη πρόσβαση μέσω RDP, η ιδιότητα PasswordNeverExpires ενεργοποιήθηκε πριν από την τελική γενίκευση του image.

Για τη γενίκευση της εγκατάστασης χρησιμοποιήθηκε το Windows Sysprep [38] σε συνδυασμό με το αρχείο:

```
C:\Windows\System32\Sysprep\unattend.xml
```

Το αρχείο απαντήσεων περιλάμβανε τις ρυθμίσεις που απαιτούνταν για την αυτοματοποιημένη αρχική διαμόρφωση του συστήματος, μεταξύ των οποίων ρυθμίσεις OOBE, τοπικοποίησης και αυτόματης σύνδεσης του Studin. Πριν από την τελική εκτέλεση του Sysprep δημιουργήθηκε snapshot της VM 108 ώστε να υπάρχει σημείο επαναφοράς της προετοιμασμένης κατάστασης. Η τελική διαδικασία γενίκευσης εκτελέστηκε με τις επιλογές:

```
/generalize /oobe /shutdown
```

και το αντίστοιχο `unattend.xml`. Μετά την ολοκλήρωση της διαδικασίας η VM τερματίστηκε, αφήνοντας το Windows golden image σε κατάσταση κατάλληλη για τη δημιουργία του τελικού deployment template.

Στην εγκατάσταση υπήρχε επίσης το αρχείο:

```
C:\Windows\Setup\Scripts\SetupComplete.cmd
```

με εντολή:

```
shutdown /s /t 0 /f
```

Η εντολή αυτή χρησιμοποιήθηκε ώστε ένα νέο Windows clone, μετά την ολοκλήρωση της πρώτης εκκίνησης και της post-Sysprep προετοιμασίας, να τερματίζεται αυτόματα. Η συμπεριφορά αυτή αξιοποιήθηκε αργότερα από το εργαλείο δημιουργίας των Windows pool VMs, το οποίο περιμένει τον αυτόματο τερματισμό πριν χαρακτηρίσει το clone ως έτοιμο.

5.3.2 Προετοιμασία του Linux Golden Image

Για το Linux χρησιμοποιήθηκε ως golden image η VMID 109, με όνομα `UbuntuServerGI`. Το guest βασίστηκε στο Ubuntu 24.04.3 LTS και προετοιμάστηκε ώστε να παρέχει γραφικό περιβάλλον μέσω απομακρυσμένης σύνδεσης RDP.

Στο σύστημα εγκαταστάθηκαν το XFCE 4.18, το `xrdp` και το `xorgxrdp`. Οι υπηρεσίες `xrdp` και `xrdp-sesman` ενεργοποιήθηκαν και το XRDP διαμορφώθηκε ώστε να δέχεται συνδέσεις στην TCP θύρα 3389. Ως τοπικός λογαριασμός του VDI περιβάλλοντος χρησιμοποιήθηκε ο χρήστης `studin`, για τον οποίο υπήρχε ενεργός κωδικός πρόσβασης.

Η έναρξη του γραφικού περιβάλλοντος για τις XRDP συνεδρίες καθορίστηκε μέσω του αρχείου:

```
/etc/xrdp/startwm.sh
```

με την ακόλουθη βασική διαμόρφωση:

```
#!/bin/sh
unset DBUS_SESSION_BUS_ADDRESS
unset XDG_RUNTIME_DIR
startxfce4
```

Με τη διαμόρφωση αυτή, μετά τη δημιουργία της Xorg-backed συνεδρίας από το XRDP, εκκινείται το περιβάλλον XFCE. Δεν χρησιμοποιήθηκε αντίστοιχο user-level `.xsession` ή `.xinitrc` για τον λογαριασμό `studin`, καθώς η εκκίνηση της συνεδρίας πραγματοποιήθηκε μέσω του system-level script του XRDP.

Στο Linux golden image εγκαταστάθηκε επίσης το QEMU Guest Agent και η υπηρεσία του ήταν ενεργή. Η παρουσία του ήταν απαραίτητη για την ίδια διαχειριστική λειτουργία που χρησιμοποιείται και στα Windows guests, δηλαδή την επικοινωνία μεταξύ Proxmox host και guest και, ειδικότερα, την ανάκτηση της IPv4 διεύθυνσης από τον broker.

Σε αντίθεση με τα Windows, η Linux εγκατάσταση δεν απαιτούσε διαδικασία Sysprep ούτε ειδικό κύκλο πρώτης εκκίνησης και αυτόματου τερματισμού. Επομένως, τα clones που προέρχονται από το τελικό Linux template μπορούν να προετοιμαστούν άμεσα για την κατάσταση `rdp-ready` από τον μηχανισμό δημιουργίας του pool.

5.3.3 Δημιουργία των Τελικών Templates και Χρήση Linked Clones

Μετά την ολοκλήρωση της προετοιμασίας των δύο golden images δημιουργήθηκαν τα templates που χρησιμοποιήθηκαν από τον τελικό lifecycle μηχανισμό.

Για το Linux, η VM 109 αποτέλεσε την πηγή του Proxmox template VMID 110, με όνομα UbuntuServerTemplate. Το template αυτό χρησιμοποιήθηκε ως βάση για τα Linux VDI clones.

Για τα Windows, από τη γενικευμένη VM 108 δημιουργήθηκε ανεξάρτητο full clone ως VMID 111, με όνομα WindowsGITemplateV3. Στη συνέχεια η VM 111 μετατράπηκε σε Proxmox template και αποτέλεσε την τελική πηγή για τα Windows VDI clones. Το παλαιότερο Windows template VMID 102 διατηρήθηκε ως εφεδρικό και δεν χρησιμοποιήθηκε για τη δημιουργία των τελικών lifecycle-managed Windows pools.

Τα VDI pool members δημιουργήθηκαν ως linked clones των αντίστοιχων templates. Η εντολή δημιουργίας που αξιοποιήθηκε από τον lifecycle μηχανισμό ακολουθεί τη μορφή[32]:

```
qm clone <template-vmid> <new-vmid>
--name <vm-name>
--full 0
```

Έτσι, το template 110 χρησιμοποιείται για τα Linux clones και το template 111 για τα Windows clones. Η περαιτέρω προετοιμασία των νέων clones, η εφαρμογή των lifecycle tags, η δημιουργία του snapshot rdp-ready και η καταχώρισή τους στα αντίστοιχα pool files πραγματοποιούνται από το εργαλείο vdi-create-pool και παρουσιάζονται αναλυτικότερα στην ενότητα διαχείρισης του κύκλου ζωής.

Η τελική αλυσίδα προετοιμασίας μπορεί επομένως να συνοψιστεί ως εξής:

- **Windows:** WindowsGI (108) → Sysprep/generalization → WindowsGITemplateV3 (111) → linked clones.
- **Linux:** UbuntuServerGI (109) → UbuntuServerTemplate (110) → linked clones.

Με τον τρόπο αυτό, οι μεταβλητές εικονικές μηχανές που χρησιμοποιήθηκαν για την προετοιμασία των λειτουργικών συστημάτων παρέμειναν διαχωρισμένες από τα templates ανάπτυξης και από τα disposable VDI pool members που δημιουργούνται και διαχειρίζονται από τα εργαλεία του συστήματος.

5.4 Υλοποίηση του Broker

Ο broker υλοποιήθηκε απευθείας στον Proxmox host και λειτουργεί ως ενδιάμεσο επίπεδο μεταξύ του thin client και των εικονικών μηχανών των VDI pools. Η λειτουργία του βασίζεται σε ένα περιορισμένο SSH entry point και σε shell scripts τα οποία αναλαμβάνουν την επιλογή διαθέσιμης εικονικής μηχανής, την εκκίνησή της και την ανάκτηση της διεύθυνσης IPv4 που επιστρέφεται στον thin client.

Η τελική αλυσίδα εκτέλεσης του broker είναι:

```
thin client
-> SSH ως lab-launcher
-> trigger_broker.sh
-> sudo ως vdi-broker
-> allocate_vm.sh <linux|windows>
-> qm / QEMU Guest Agent
-> IPv4 προς thin client
```

5.4.1 Περιορισμένο SSH Entry Point και Διαχωρισμός Δικαιωμάτων

Για την αποστολή αιτημάτων από τον thin client δημιουργήθηκε στον Proxmox host ο λογαριασμός lab-launcher. Η αυθεντικοποίηση πραγματοποιείται με SSH key και το αντίστοιχο κλειδί στο αρχείο `authorized_keys` συνδέθηκε με `forced command`:

```
command="/usr/local/bin/trigger_broker.sh",
no-port-forwarding,no-agent-forwarding,
no-X11-forwarding,no-pty
```

Με αυτή τη διαμόρφωση, μία επιτυχής SSH σύνδεση με το συγκεκριμένο κλειδί δεν παρέχει διαδραστικό shell στον thin client. Επιπλέον, απενεργοποιούνται το port forwarding, το agent forwarding, το X11 forwarding και η δημιουργία PTY. Το αίτημα οδηγείται υποχρεωτικά στο `/usr/local/bin/trigger_broker.sh`[48].

Ο thin client αποστέλλει ως SSH command μόνο το όνομα του επιλεγμένου pool, δηλαδή linux ή windows. Το `trigger_broker.sh` διαβάζει την τιμή από τη μεταβλητή `SSH_ORIGINAL_COMMAND` και απορρίπτει τόσο την απουσία επιλογής όσο και οποιαδήποτε τιμή εκτός των δύο υποστηριζόμενων pools. Η βασική λογική ελέγχου είναι:

```
POOL="${SSH_ORIGINAL_COMMAND:-}"

case "$POOL" in
linux|windows)
;;
"")
echo "ERROR: No VDI pool specified." >&2
exit 1
;;
*)
echo "ERROR: Invalid VDI pool requested." >&2
exit 1
;;
esac
```

Μετά την επικύρωση της επιλογής, το script εκτελεί τον allocator ως διαφορετικό λογαριασμό υπηρεσίας:

```
exec sudo -n -u vdi-broker
/usr/local/bin/allocate_vm.sh "$POOL"
```

Ο λογαριασμός `vdi-broker` χρησιμοποιείται επομένως ως ξεχωριστή ταυτότητα εκτέλεσης του allocator. Μέσω των αντίστοιχων κανόνων `sudoers`, ο `lab-launcher` μπορεί να εκτελεί τον allocator ως `vdi-broker` χωρίς εισαγωγή κωδικού, ενώ ο `vdi-broker` διαθέτει τις απαιτούμενες εξουσιοδοτήσεις για συγκεκριμένες λειτουργίες του `qm`, όπως έλεγχο κατάστασης, εκκίνηση VM και επικοινωνία με το QEMU Guest Agent.

5.4.2 Επιλογή Pool και Ανάγνωση Διαθέσιμων VMIDs

Ο allocator υλοποιήθηκε στο αρχείο:

```
/usr/local/bin/allocate_vm.sh
```

και υποστηρίζει την επιλογή διαφορετικού pool μέσω argument. Τα αρχεία των pools βρίσκονται στον κατάλογο:

```
/etc/vdi-broker/pools/
```

Στην τελική διαμόρφωση χρησιμοποιούνται τα:

```
/etc/vdi-broker/pools/linux.pool  
/etc/vdi-broker/pools/windows.pool
```

τα οποία περιέχουν τα VMIDs των εικονικών μηχανών που είναι καταχωρισμένες στο αντίστοιχο pool. Για παράδειγμα, η τελική πειραματική διαμόρφωση περιλάμβανε:

```
# linux.pool  
  
120  
121  
  
# windows.pool  
  
130  
131
```

Το όνομα του pool μετατρέπεται από τον allocator στο αντίστοιχο αρχείο:

```
PPOOL_NAME="${1:-default}"  
PPOOL_FILE="${PPOOL_DIR}/${PPOOL_NAME}.pool"
```

Ο allocator ελέγχει ότι το αρχείο είναι αναγνώσιμο και φορτώνει μόνο τις γραμμές που περιέχουν VMIDs, αγνοώντας κενές γραμμές ή σχόλια. Με αυτόν τον τρόπο ο ίδιος μηχανισμός allocation μπορεί να χρησιμοποιηθεί για διαφορετικές κατηγορίες λειτουργικών συστημάτων χωρίς να απαιτείται διαφορετικό script για κάθε pool.

5.4.3 Επιλογή και Εκκίνηση Διαθέσιμης Εικονικής Μηχανής

Μετά τη φόρτωση του pool, το `allocate_vm.sh` εξετάζει διαδοχικά τα καταχωρισμένα VMIDs αναζητώντας μία διαθέσιμη εικονική μηχανή. Για κάθε υποψήφια VM χρησιμοποιείται μη αποκλειστικό ανά VM lock στον κοινό κατάλογο:

```
/var/lock/vdi-broker/
```

με αρχεία της μορφής:

```
vm-<VMID>.lock
```

Η απόκτηση του lock επιχειρείται με `flock -n`. Εάν το συγκεκριμένο VMID χρησιμοποιείται ήδη από άλλη συνεργαζόμενη λειτουργία, ο allocator το παρακάμπτει και συνεχίζει με το επόμενο[51].

Αφού αποκτήσει το lock, ο allocator επανελέγχει την κατάσταση της VM μέσω του `qm status`. Η επιλογή πραγματοποιείται μόνο εφόσον η VM βρίσκεται σε κατάσταση `stopped`. Ο επανέλεγχος γίνεται ενώ το lock παραμένει ενεργό, ώστε η επιλογή να βασίζεται στην τρέχουσα κατάσταση της VM και όχι σε προηγούμενο έλεγχο.

Η βασική λογική μπορεί να αποδοθεί συνοπτικά ως:

```
for each VMID in pool:
  try non-blocking VM lock
  if lock acquired:
    re-check VM state
    if state == stopped:
      select VM
      break
```

Εάν δεν εντοπιστεί διαθέσιμη VM, το αίτημα τερματίζεται με μήνυμα αποτυχίας προς τον thin client. Εάν βρεθεί κατάλληλη VM, ο allocator την εκκινεί μέσω:

```
qm start <VMID>
```

Το lock της επιλεγμένης VM διατηρείται κατά τη διαδικασία εκκίνησης και μέχρι να ολοκληρωθεί η ανάκτηση και επιστροφή της διεύθυνσης IP.

5.4.4 Ανάκτηση της IPv4 Διεύθυνσης μέσω QEMU Guest Agent

Η εκκίνηση της VM δεν αρκεί για την άμεση έναρξη της σύνδεσης RDP, καθώς ο thin client χρειάζεται πρώτα τη δικτυακή της διεύθυνση. Για τον σκοπό αυτό ο allocator χρησιμοποιεί το QEMU Guest Agent που έχει εγκατασταθεί στα Windows και Linux images.

Μετά την εκκίνηση, πραγματοποιούνται επαναλαμβανόμενα αιτήματα της μορφής:

```
qm guest cmd <VMID> network-get-interfaces
```

Η απάντηση του guest agent περιέχει τις δικτυακές διεπαφές και τις διευθύνσεις τους. Το αποτέλεσμα επεξεργάζεται με `jq` και επιλέγεται η πρώτη κατάλληλη IPv4 διεύθυνση[36]. Από την επιλογή αποκλείονται οι loopback διευθύνσεις `127.*` και οι link-local διευθύνσεις `169.254.*`.

Ο allocator περιμένει έως και 120 δευτερόλεπτα για την εμφάνιση κατάλληλης IPv4 διεύθυνσης, πραγματοποιώντας νέο έλεγχο ανά 5 δευτερόλεπτα. Η καθυστέρηση αυτή επιτρέπει στο guest λειτουργικό σύστημα και στο QEMU Guest Agent να ολοκληρώσουν την εκκίνησή τους χωρίς ο thin client να χρειάζεται να γνωρίζει ή να εφαρμόζει διαφορετικό μηχανισμό αναμονής για Windows και Linux.

Όταν εντοπιστεί κατάλληλη διεύθυνση, αυτή αποτελεί το αποτέλεσμα του αιτήματος allocation και επιστρέφεται μέσω της αρχικής SSH σύνδεσης στον thin client. Το lock της VM απελευθερώνεται μετά την επιτυχή επιστροφή της διεύθυνσης.

5.4.5 Έξοδος προς τον Client, Καταγραφή και Χειρισμός Αποτυχιών

Το `allocate_vm.sh` διαχωρίζει την έξοδο που προορίζεται για τον thin client από τα διαγνωστικά μηνύματα του broker. Κατά την έναρξη του script διατηρείται αντίγραφο του αρχικού `stdout` σε ξεχωριστό file descriptor, ενώ η κανονική διαγνωστική έξοδος ανακατευθύνεται στο:

```
/var/log/vdi-broker/allocate_vm.log
```

Έτσι, τα μηνύματα που αφορούν την αναζήτηση VM, την κατάσταση του guest agent και την εσωτερική λειτουργία του allocator δεν αναμειγνύονται με την τιμή που πρέπει να επεξεργαστεί ο thin client. Σε περίπτωση επιτυχίας επιστρέφεται μόνο η IPv4 διεύθυνση, ενώ σε περίπτωση αποτυχίας επιστρέφεται κατάλληλο μήνυμα σφάλματος.

Το `trigger_broker.sh` πραγματοποιεί επιπλέον καταγραφή της ενεργοποίησής του μέσω του logger, χρησιμοποιώντας το tag `vdi-trigger`. Η καταγραφή περιλαμβάνει πληροφορίες για τον χρόνο ενεργοποίησης, τον χρήστη, τη σύνδεση SSH και την τιμή του `SSH_ORIGINAL_COMMAND`.

Οι βασικές περιπτώσεις αποτυχίας που αντιμετωπίζονται από τον broker περιλαμβάνουν μη έγκυρο ή ανύπαρκτο pool, κενό pool, απουσία διαθέσιμης stopped VM, αποτυχία εκκίνησης της επιλεγμένης VM και υπέρβαση του χρονικού ορίου αναμονής για την απόκτηση IPv4 διεύθυνσης.

Η υλοποίηση αυτή αφήνει στον thin client μία περιορισμένη διεπαφή: αποστέλλει μόνο την επιλογή `linux` ή `windows` και λαμβάνει ως απάντηση τη διεύθυνση της εικονικής μηχανής που θα χρησιμοποιήσει. Οι λεπτομέρειες επιλογής, εκκίνησης και επικοινωνίας με το Proxmox/QEMU περιβάλλον παραμένουν στον broker.

5.5 Υλοποίηση του Thin Client

Ο thin client υλοποιήθηκε ως ελαφρύ Debian 12 σύστημα, με στόχο η διαδικασία από την εκκίνηση μέχρι τη σύνδεση σε διαθέσιμη VDI εικονική μηχανή να απαιτεί την ελάχιστη δυνατή αλληλεπίδραση από τον χρήστη. Στην τελική διαμόρφωση ο χρήστης δεν χρειάζεται να πραγματοποιήσει χειροκίνητη είσοδο στο λειτουργικό σύστημα ούτε να εκτελέσει εντολές για την

έναρξη του γραφικού περιβάλλοντος. Η μόνη απαιτούμενη επιλογή είναι το επιθυμητό VDI περιβάλλον, Linux ή Windows.

Η τελική ροή του thin client είναι:

```
tty1 autologin
-> vdi-menu.sh
-> επιλογή Linux ή Windows
-> startx
-> .xinitrc
-> vdi-rdp.sh
-> SSH request προς broker
-> IPv4 VM
-> FreeRDP
```

5.5.1 Αυτόματη Είσοδος και Επιλογή Λειτουργικού Συστήματος

Στην κύρια κονσόλα `tty1` διαμορφώθηκε αυτόματη είσοδος του τοπικού χρήστη `papas` μέσω `systemd override` της υπηρεσίας `getty`. Η βασική ρύθμιση είναι:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin papas
--noclear %I $TERM
```

Μετά την αυτόματη είσοδο, το αρχείο `$HOME/.profile` ελέγχει εάν η συνεδρία εκτελείται στην `/dev/tty1`. Στην περίπτωση αυτή εκκινείται το script επιλογής του VDI περιβάλλοντος:

```
if [ "$(tty 2>/dev/null)" = "/dev/tty1" ]; then
exec "$HOME/bin/vdi-menu.sh"
fi
```

Ο έλεγχος της συγκεκριμένης κονσόλας αποτρέπει την αυτόματη εκκίνηση του VDI workflow σε άλλες τοπικές συνεδρίες.

Το `vdi-menu.sh` εμφανίζει ένα απλό text-based menu με δύο επιλογές:

1. Linux
2. Windows

Ανάλογα με την επιλογή του χρήστη, ορίζεται η μεταβλητή περιβάλλοντος `VDI_POOL`:

```
VDI_POOL=linux
```

ή:

```
VDI_POOL=windows
```

Η μεταβλητή εξάγεται στο περιβάλλον της διεργασίας και στη συνέχεια εκτελείται το `startx`. Με αυτόν τον τρόπο η επιλογή που πραγματοποιήθηκε πριν από την εκκίνηση του X environment μεταφέρεται στο script που θα πραγματοποιήσει το αίτημα προς τον broker.

5.5.2 Εκκίνηση της Γραφικής Συνεδρίας και Σύνδεση μέσω FreeRDP

Η γραφική συνεδρία του thin client διαμορφώθηκε ώστε να εξυπηρετεί άμεσα τη VDI σύνδεση. Στο αρχείο:

```
/home/papas/.xinitrc
```

απενεργοποιούνται αρχικά οι μηχανισμοί screen blanking και power management:

```
xset s off
xset -dpms
xset s noblank
```

και στη συνέχεια εκτελείται το κύριο script σύνδεσης:

```
exec "$HOME/bin/vdi-rdp.sh"
```

Το `vdi-rdp.sh` ελέγχει αρχικά ότι η μεταβλητή `VDI_POOL` έχει μία από τις δύο επιτρεπόμενες τιμές, `linux` ή `windows`. Εάν η τιμή απουσιάζει ή δεν είναι έγκυρη, το αίτημα δεν αποστέλλεται στον broker.

Για την επικοινωνία με τον broker χρησιμοποιείται SSH προς τον περιορισμένο λογαριασμό:

```
lab-launcher@192.168.6.35
```

με το αντίστοιχο ιδιωτικό κλειδί του thin client. Η επιλεγμένη τιμή του `VDI_POOL` αποστέλλεται ως SSH command:

```
ssh ... "$BROKER" "$VDI_POOL"
```

Όπως περιγράφηκε στην προηγούμενη ενότητα, το SSH key του `lab-launcher` είναι συνδεδεμένο με `forced command` στον Proxmox host. Επομένως, η τιμή `linux` ή `windows` δεν εκτελείται ως αυθαίρετη εντολή κελύφους αλλά παραδίδεται στο `trigger_broker.sh`, το οποίο την επικυρώνει και ενεργοποιεί τον αντίστοιχο allocator.

Το αποτέλεσμα της SSH σύνδεσης αποθηκεύεται από το `vdi-rdp.sh` ως η διεύθυνση της VM:

```
IP=$(ssh ... "$BROKER" "$VDI_POOL")
```

Πριν χρησιμοποιηθεί η απάντηση, το script ελέγχει ότι δεν είναι κενή ή μήνυμα σφάλματος και ότι ακολουθεί τη μορφή IPv4 διεύθυνσης. Με τον τρόπο αυτό μία αποτυχία του broker δεν μεταφέρεται αυτούσια ως παράμετρος στον FreeRDP client.

Εφόσον επιστραφεί έγκυρη διεύθυνση, εκκινείται το FreeRDP σε πλήρη οθόνη:

```
xfreerdp
/v:"$IP"
/u:"$USER"
/p:"$PASS"
/f
/cert:ignore
/network:lan
```

Η ίδια client-side διαδικασία χρησιμοποιείται τόσο για Windows όσο και για Linux VMs. Στην πρώτη περίπτωση η σύνδεση καταλήγει στην υπηρεσία Remote Desktop των Windows, ενώ στη δεύτερη στο XRDP service του Linux guest. Ο thin client επομένως δεν χρειάζεται διαφορετικό RDP workflow ανά λειτουργικό σύστημα πέρα από την αρχική επιλογή του αντίστοιχου pool.

5.5.3 Χειρισμός Αποτυχιών και Τοπική Καταγραφή

Το `vdi-rdp.sh` διατηρεί τοπικό αρχείο καταγραφής:

```
~/vdi-rdp.log
```

στο οποίο καταγράφονται η έναρξη του script, το pool που ζητήθηκε, η απάντηση του broker και πληροφορίες σχετικές με την έναρξη της σύνδεσης.

Για την SSH επικοινωνία χρησιμοποιήθηκαν επίσης επιλογές κατάλληλες για μη διαδραστική λειτουργία, όπως:

```
BatchMode=yes
ConnectTimeout=15
ServerAliveInterval=10
ServerAliveCountMax=2
```

ώστε η διαδικασία να μην περιμένει επ' αόριστον για διαδραστική είσοδο ή για μία μη λειτουργική σύνδεση.

Εάν ο broker επιστρέψει κενή απάντηση ή μήνυμα σφάλματος, ή εάν η απάντηση δεν αντιστοιχεί στη μορφή IPv4 διεύθυνσης, το script εμφανίζει μήνυμα σφάλματος μέσω `xmessage` και δεν επιχειρεί σύνδεση FreeRDP. Στην υλοποίηση χρησιμοποιείται στη συνέχεια ένα `xterm`, παρέχοντας τοπικό περιβάλλον μετά την αποτυχία αντί να επιχειρείται σύνδεση με μη έγκυρο προορισμό.

Η τελική υλοποίηση του thin client περιορίζει επομένως τη διαδικασία που βλέπει ο τελικός χρήστης στην επιλογή του επιθυμητού λειτουργικού περιβάλλοντος. Η αναζήτηση διαθέσιμης VM, η εκκίνησή της και η ανακάλυψη της δικτυακής της διεύθυνσης πραγματοποιούνται από τον broker, ενώ ο thin client αναλαμβάνει την επιλογή του pool, την αποστολή του αιτήματος και την έναρξη της τελικής RDP σύνδεσης.

5.6 Υλοποίηση Διαχείρισης Κύκλου Ζωής

Πέρα από την ανάθεση εικονικών μηχανών μέσω του broker, υλοποιήθηκε ένα σύνολο εργαλείων για τη διαχείριση του κύκλου ζωής των VDI pools. Τα εργαλεία αυτά εκτελούνται στον Proxmox host και καλύπτουν τη δημιουργία και αναπλήρωση των pools, την επαναφορά των εικονικών μηχανών σε προκαθορισμένη καθαρή κατάσταση, την ελεγχόμενη καταστροφή τους και την εποπτεία της τρέχουσας κατάστασης.

Η τελική διάταξη περιλαμβάνει τα ακόλουθα εργαλεία:

```
/usr/local/sbin/vdi-create-pool
/usr/local/sbin/vdi-reset-pool
/usr/local/sbin/vdi-destroy-pool
/usr/local/sbin/vdi-status
```

Τα scripts χρησιμοποιούν κοινή διαμόρφωση και κοινά αρχεία καταχώρισης των pools, ώστε τα όρια των διαχειριζόμενων VMs και η επιθυμητή κατάσταση του συστήματος να μην ορίζονται ανεξάρτητα σε κάθε εργαλείο.

5.6.1 Κοινή Διαμόρφωση και Pool Files

Οι βασικές παράμετροι του lifecycle management συγκεντρώθηκαν στο αρχείο:

```
/etc/vdi-broker/lifecycle.conf
```

Η τελική διαμόρφωση περιλαμβάνει:

```
LINUX_TEMPLATE=110
WINDOWS_TEMPLATE=111

LINUX_POOL_SIZE=2
WINDOWS_POOL_SIZE=2

LINUX_VMID_START=120
LINUX_VMID_END=129
WINDOWS_VMID_START=130
WINDOWS_VMID_END=139

LINUX_NAME_PREFIX=linux-vdi
WINDOWS_NAME_PREFIX=windows-vdi

LINUX_POOL_FILE=/etc/vdi-broker/pools/linux.pool
WINDOWS_POOL_FILE=/etc/vdi-broker/pools/windows.pool

READY_SNAPSHOT=rdp-ready
READY_TAG=rdp-ready
RESET_MIN_UPTIME=300
```

Με τον τρόπο αυτό ορίζονται κεντρικά το template κάθε λειτουργικού συστήματος, το επιθυμητό μέγεθος του pool, η περιοχή VMIDs μέσα στην οποία επιτρέπεται η δημιουργία διαχειριζόμενων clones, η ονοματοδοσία τους, το αντίστοιχο pool file και οι παράμετροι που χαρακτηρίζουν μία VM ως έτοιμη για χρήση.

Τα αρχεία `linux.pool` και `windows.pool` λειτουργούν ως μητρώα των VMs που έχουν ολοκληρώσει επιτυχώς τη διαδικασία προετοιμασίας και μπορούν να χρησιμοποιηθούν από τον broker. Στην τελική πειραματική διαμόρφωση περιείχαν:

```
# linux.pool
```

```
120
121
```

```
# windows.pool
```

```
130
```

```
131
```

Η απλή ύπαρξη μιας VM μέσα στη δεσμευμένη περιοχή VMID δεν την καθιστά αυτόματα μέλος του pool. Η καταχώρισή της πραγματοποιείται μόνο αφού ολοκληρωθούν επιτυχώς τα απαιτούμενα βήματα προετοιμασίας.

5.6.2 Δημιουργία και Αναπλήρωση Pool με το vdi-create-pool

Το `vdi-create-pool` υλοποιήθηκε ως εργαλείο δημιουργίας και διατήρησης της επιθυμητής χωρητικότητας ενός Linux ή Windows pool. Η χρήση του ακολουθεί τη μορφή:

```
vdi-create-pool {linux|windows}
```

Με βάση την επιλογή του λειτουργικού συστήματος, το script φορτώνει από το `lifecycle.conf` το κατάλληλο template, το επιθυμητό μέγεθος του pool, τη δεσμευμένη περιοχή VMIDs, το prefix ονοματοδοσίας και το αντίστοιχο pool file.

Αρχικά διαβάζονται τα VMIDs που είναι ήδη καταχωρισμένα στο pool και ελέγχεται ποια από αυτά εξακολουθούν να αντιστοιχούν σε υπαρκτές Proxmox VMs. Στη συνέχεια υπολογίζεται η διαφορά μεταξύ του αριθμού των έγκυρων υφιστάμενων μελών και του επιθυμητού μεγέθους:

```
DEFICIT = TARGET_SIZE - VALID_COUNT
```

Εάν το pool βρίσκεται ήδη στο επιθυμητό μέγεθος, το εργαλείο τερματίζεται χωρίς να δημιουργήσει νέες VMs. Διαφορετικά αναζητά ελεύθερα VMIDs αποκλειστικά μέσα στη δεσμευμένη περιοχή του επιλεγμένου λειτουργικού συστήματος και δημιουργεί μόνο τον αριθμό των clones που απαιτείται για την κάλυψη του ελλείμματος.

Η δημιουργία πραγματοποιείται ως linked clone του αντίστοιχου template. Σε γενική μορφή χρησιμοποιείται:

```
qm clone <template> <vmid>
--name <generated-name>
--full 0
```

Η περαιτέρω προετοιμασία διαφοροποιείται μεταξύ Linux και Windows. Για Linux δεν απαιτείται ειδικός κύκλος πρώτης εκκίνησης και το νέο clone μπορεί να προχωρήσει άμεσα στα lifecycle metadata και στο καθαρό snapshot.

Για Windows, αντίθετα, το νέο clone πρέπει να ολοκληρώσει την πρώτη post-Sysprep εκκίνησή του. Το `vdi-create-pool` εκκινεί τη VM και περιμένει τον αυτόματο τερματισμό που προκαλείται από το `SetupComplete.cmd`. Μόνο μετά την ολοκλήρωση αυτού του κύκλου συνεχίζεται η προετοιμασία.

Η τελική σειρά ενεργειών για ένα επιτυχώς προετοιμασμένο clone είναι:

```
clone
-> OS-specific preparation
-> lifecycle tags
-> rdp-ready snapshot
-> pool registration
```

Τα lifecycle tags περιλαμβάνουν το γενικό tag `vdi`, το λειτουργικό σύστημα και το `rdp-ready`. Ενδεικτικά, οι τελικές μορφές είναι:

```
linux;rdp-ready;vdi
rdp-ready;vdi;windows
```

Τα tags εφαρμόζονται πριν από τη δημιουργία του snapshot `rdp-ready`. Η σειρά αυτή είναι σημαντική επειδή το Proxmox snapshot περιλαμβάνει και κατάσταση της διαμόρφωσης της VM[32]. Επομένως, η επαναφορά σε snapshot πρέπει να επαναφέρει και τα σωστά lifecycle metadata.

Η εγγραφή του VMID στο αντίστοιχο pool file αποτελεί το τελευταίο βήμα. Έτσι, ένα clone που απέτυχε σε προηγούμενο στάδιο της προετοιμασίας δεν γίνεται διαθέσιμο στον allocator ως κανονικό μέλος του pool.

Το ίδιο εργαλείο χρησιμοποιείται και για αναπλήρωση. Εάν ένα διαχειριζόμενο clone αφαιρεθεί και το pool αποκτήσει μικρότερο αριθμό έγκυρων μελών από το `POOL_SIZE`, μία επόμενη εκτέλεση του `vdi-create-pool` εντοπίζει το έλλειμμα και δημιουργεί μόνο τα μέλη που λείπουν.

5.6.3 Επαναφορά Καθαρής Κατάστασης με το `vdi-reset-pool`

Για την επαναφορά των disposable VDI clones υλοποιήθηκε το:

```
/usr/local/sbin/vdi-reset-pool
```

Το εργαλείο μπορεί να εφαρμοστεί σε ολόκληρο Linux ή Windows pool ή σε συγκεκριμένο VMID:

```
vdi-reset-pool {linux|windows} [VMID]
```

Πριν από οποιαδήποτε επαναφορά πραγματοποιούνται διαδοχικοί έλεγχοι ώστε η λειτουργία να εφαρμοστεί μόνο σε VM που ανήκει πράγματι στο διαχειριζόμενο VDI σύνολο. Ελέγχονται, μεταξύ άλλων, η καταχώριση στο επιλεγμένο pool, η ύπαρξη της VM, η ένταξή της στην επιτρεπόμενη περιοχή VMID, η παρουσία των απαιτούμενων lifecycle tags και η ύπαρξη του snapshot `rdp-ready`.

Για κάθε VM χρησιμοποιείται το ίδιο ανά-VM lock namespace με τον allocator. Εάν η VM χρησιμοποιείται ήδη από άλλη συνεργαζόμενη λειτουργία, το reset δεν προχωρά ταυτόχρονα πάνω στον ίδιο πόρο.

Οι stopped VMs μπορούν να επανέλθουν απευθείας στο `rdp-ready` snapshot. Για μία running VM εφαρμόζονται πρόσθετοι έλεγχοι πριν επιτραπεί η επαναφορά.

Αρχικά χρησιμοποιείται η παράμετρος:

```
RESET_MIN_UPTIME=300
```

ώστε μία VM που έχει μόλις εκκινηθεί να μην επαναφερθεί κατά τη διάρκεια του χρονικού διαστήματος στο οποίο ο broker περιμένει το QEMU Guest Agent, επιστρέφει την IP και ο thin client επιχειρεί να δημιουργήσει την RDP σύνδεση.

Εφόσον έχει παρέλθει το ελάχιστο uptime, πραγματοποιείται έλεγχος για ενεργή απομακρυσμένη συνεδρία. Στα Windows χρησιμοποιείται αρχικά το `quser`. Ως δεύτερος έλεγχος εκτελείται PowerShell query για ενεργή TCP σύνδεση στην τοπική θύρα 3389, το οποίο επιστρέφει τον μοναδικό δείκτη:

```
ACTIVE_RDP
```

μόνο όταν υπάρχει πραγματικά established RDP connection.

Στα Linux guests ο έλεγχος πραγματοποιείται μέσω:

```
ss -tnp
```

αναζητώντας established σύνδεση στην τοπική TCP θύρα 3389 του XRDP.

Εάν εντοπιστεί ενεργή συνεδρία, η VM παραλείπεται. Εάν δεν εντοπιστεί δραστηριότητα, πραγματοποιείται ένας τελικός επανέλεγχος πριν από τη μεταβολή της κατάστασης. Στη συνέχεια η VM τερματίζεται, εφόσον απαιτείται, και εκτελείται rollback:

```
qm rollback <VMID> rdp-ready
```

Μετά την επαναφορά επαληθεύεται ότι η VM βρίσκεται στην αναμενόμενη stopped κατάσταση και εξακολουθεί να διαθέτει τα απαιτούμενα tags και snapshot. Με αυτόν τον τρόπο το ίδιο εργαλείο μπορεί να χρησιμοποιηθεί τόσο για μεμονωμένη συντήρηση όσο και για περιοδική επαναφορά ενός ολόκληρου pool, χωρίς να επαναφέρει σκόπιμα μία VM στην οποία έχει εντοπιστεί ενεργή συνεδρία.

Η ακολουθία των ελέγχων που προηγούνται της επαναφοράς μιας εικονικής μηχανής παρουσιάζεται συνοπτικά στο Σχήμα 5.1.

5.6.4 Ελεγχόμενη Καταστροφή με το vdi-destroy-pool

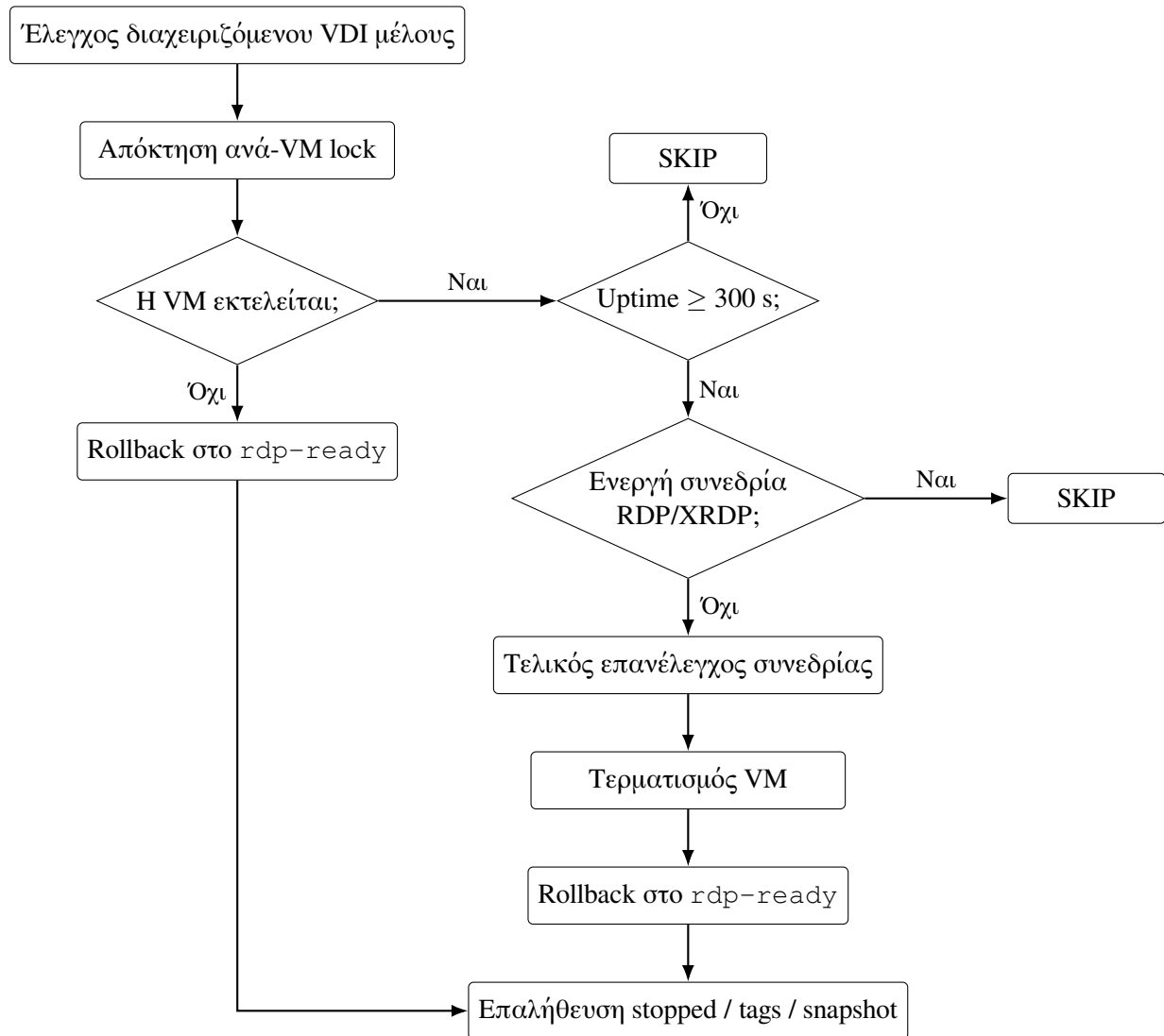
Για την αφαίρεση disposable VDI clones υλοποιήθηκε το:

```
/usr/local/sbin/vdi-destroy-pool
```

Η τελική έκδοση υποστηρίζει δύο μορφές χρήσης:

```
vdi-destroy-pool {linux|windows} VMID
vdi-destroy-pool {linux|windows}
```

Στην πρώτη περίπτωση στοχεύεται ένα συγκεκριμένο μέλος του pool. Στη δεύτερη περίπτωση στοχεύονται όλα τα VMIDs που είναι καταχωρισμένα στο επιλεγμένο pool. Η λειτουργία ολόκληρου pool δεν παρακάμπτει τους ελέγχους προστασίας· κάθε VM εξετάζεται και διαχειρίζεται ανεξάρτητα.



Σχήμα 5.1: Λογική ελέγχου και ασφαλούς επαναφοράς εικονικής μηχανής.

Το script διαβάσει αρχικά ένα snapshot της τρέχουσας λίστας VMIDs του pool. Εάν έχει δοθεί συγκεκριμένο VMID, απαιτείται αυτό να υπάρχει ήδη στη συγκεκριμένη λίστα. Εάν δεν έχει δοθεί VMID, η αρχική λίστα χρησιμοποιείται ως το σύνολο στόχων για την τρέχουσα λειτουργία.

Για κάθε υποψήφια VM εφαρμόζονται περιορισμοί ώστε να μην μπορεί το εργαλείο να χρησιμοποιηθεί ως γενικός μηχανισμός διαγραφής αυθαίρετων Proxmox VMs. Ο στόχος πρέπει να βρίσκεται στη δεσμευμένη περιοχή VMID του επιλεγμένου λειτουργικού συστήματος και να είναι καταχωρισμένος στο αντίστοιχο pool. Επιπλέον ελέγχονται τα lifecycle tags και απορρίπτονται προστατευμένες καταστάσεις, όπως Proxmox templates ή VMs που φέρουν το tag `gi`. Πριν από την καταστροφή αποκτάται το κοινό per-VM lifecycle lock. Εφόσον ο στόχος είναι running, τερματίζεται με ελεγχόμενο τρόπο πριν εκτελεστεί η καταστροφή. Η αφαίρεση πραγματοποιείται μέσω του `qm destroy` και το VMID διαγράφεται από το pool file μόνο αφού η καταστροφή ολοκληρωθεί επιτυχώς.

Η σειρά των ενεργειών μπορεί να συνοψιστεί ως:

```
pool membership
-> managed-range validation
-> lifecycle/protection checks
-> per-VM lock
-> stop if required
-> qm destroy
-> remove VMID from pool file
```

Στη λειτουργία ολόκληρου pool η ίδια διαδικασία επαναλαμβάνεται για κάθε καταχωρισμένο μέλος. Έτσι, η μαζική λειτουργία αποτελεί επέκταση της προστατευμένης ανά-VM καταστροφής και όχι διαφορετική ανεξέλεγκτη διαδρομή.

Μετά την αφαίρεση ενός ή περισσότερων clones, το `vdi-create-pool` μπορεί να χρησιμοποιηθεί για την αποκατάσταση της επιθυμητής χωρητικότητας. Ο συνδυασμός των δύο εργαλείων επιτρέπει επομένως τόσο την ελεγχόμενη εκκαθάριση ενός pool όσο και τη μεταγενέστερη αναδημιουργία των disposable μελών του από τα αντίστοιχα templates.

5.6.5 Εποπτεία της Κατάστασης με το `vdi-status`

Για τη συγκεντρωτική εποπτεία της κατάστασης των VDI pools υλοποιήθηκε το:

```
/usr/local/sbin/vdi-status
```

Σε αντίθεση με τα προηγούμενα lifecycle εργαλεία, το `vdi-status` έχει αποκλειστικά εποπτικό χαρακτήρα και δεν μεταβάλλει την κατάσταση των εικονικών μηχανών.

Για κάθε λειτουργικό σύστημα εμφανίζει τον αριθμό των καταχωρισμένων VMs σε σχέση με το επιθυμητό μέγεθος του pool και τη δεσμευμένη περιοχή VMID. Για κάθε μέλος παρουσιάζονται πληροφορίες όπως:

- VMID και όνομα,
- κατάσταση running ή stopped,

- uptime, όπου εφαρμόζεται,
- κατάσταση RDP/XRDP,
- παρουσία του snapshot rdp-ready,
- εγκυρότητα των lifecycle tags,
- κατάσταση του αντίστοιχου per-VM lock.

Η έξοδος έχει τη γενική μορφή:

VMID	NAME	STATE	UPTIME	RDP	SNAPSHOT	TAGS	LOCK
120	linux-vdi-01	stopped	-	n/a	ready	OK	free
121	linux-vdi-02	stopped	-	n/a	ready	OK	free

Η τελική υλοποίηση του ελέγχου των locks διατηρεί τον read-only χαρακτήρα του εργαλείου: εάν το αντίστοιχο lock file δεν υπάρχει, η κατάσταση αναφέρεται ως ελεύθερη χωρίς το vdi-status να δημιουργεί νέο αρχείο lock.

Με τα τέσσερα παραπάνω εργαλεία σχηματίζεται ένας ενιαίος μηχανισμός διαχείρισης των disposable VDI πόρων. Το vdi-create-pool δημιουργεί ή αναπληρώνει την επιθυμητή χωρητικότητα, το vdi-reset-pool επαναφέρει ασφαλώς κατάλληλα μέλη στην καθαρή κατάσταση rdp-ready, το vdi-destroy-pool αφαιρεί ελεγχόμενα μεμονωμένα ή όλα τα μέλη ενός pool και το vdi-status παρέχει συγκεντρωτική εικόνα της τρέχουσας κατάστασης.

5.7 Τελική Ολοκλήρωση των Υποσυστημάτων

Μετά την υλοποίηση του broker, του thin client και των εργαλείων διαχείρισης του κύκλου ζωής, πραγματοποιήθηκε η τελική ολοκλήρωσή τους σε μία κοινή λειτουργική ροή. Σε αυτό το στάδιο ιδιαίτερη σημασία είχαν οι μηχανισμοί που χρησιμοποιούνται από περισσότερα του ενός υποσυστήματα, κυρίως ο συγχρονισμός των λειτουργιών πάνω στις ίδιες εικονικές μηχανές και ο διαχωρισμός της λειτουργικής εξόδου από τα διαγνωστικά δεδομένα.

5.7.1 Κοινός Μηχανισμός Κλειδώματος

Ο allocator και τα lifecycle εργαλεία μπορούν να επιχειρήσουν διαφορετικές ενέργειες πάνω στην ίδια VM. Για παράδειγμα, ένα αίτημα allocation μπορεί να συμπέσει χρονικά με μία διαδικασία reset ή καταστροφής. Για τον συντονισμό αυτών των λειτουργιών χρησιμοποιήθηκε κοινός μηχανισμός per-VM locking μέσω flock.

Όλα τα συνεργαζόμενα εργαλεία χρησιμοποιούν τον κοινό κατάλογο:

```
/var/lock/vdi-broker/
```

και ένα lock file ανά VM:

```
/var/lock/vdi-broker/vm-<VMID>.lock
```

Η απόκτηση πραγματοποιείται χωρίς αναμονή μέσω `flock -n`. Εάν το lock μιας VM κατέχεται ήδη από άλλη λειτουργία, η δεύτερη λειτουργία δεν προχωρά ταυτόχρονα πάνω στον ίδιο πόρο. Ο allocator μπορεί να συνεχίσει την αναζήτηση άλλου διαθέσιμου μέλους του pool, ενώ λειτουργίες όπως `reset` ή `destroy` μπορούν να απορρίψουν ή να παραλείψουν τον συγκεκριμένο στόχο.

Επειδή ο allocator εκτελείται ως `vdi-broker`, ενώ τα lifecycle εργαλεία εκτελούνται με διαχειριστικά δικαιώματα, ο κοινός κατάλογος των locks διαμορφώθηκε ώστε να είναι προσβάσιμος και από τις δύο πλευρές. Η τελική διαμόρφωση του καταλόγου είναι:

```
owner: root
group: vdi-broker
mode: 2770
```

Το set-GID bit διατηρεί την ομάδα `vdi-broker` στα αντικείμενα που δημιουργούνται μέσα στον κατάλογο. Παράλληλα, στα scripts που μπορούν να δημιουργήσουν lock files χρησιμοποιείται:

```
umask 0007
```

ώστε τα νέα αρχεία να παραμένουν εγγράψιμα από την κοινή ομάδα.

Για τη διατήρηση της σωστής κατάστασης και μετά από επανεκκίνηση του host προστέθηκε κανόνας `systemd-tmpfiles` για τον runtime κατάλογο `/run/lock/vdi-broker`, στον οποίο αντιστοιχεί το `/var/lock/vdi-broker`. Έτσι ο κατάλογος επαναδημιουργείται με τα προβλεπόμενα ownership και permissions αντί να εξαρτάται από το ποιο script θα εκτελεστεί πρώτο.

Τα lifecycle scripts διαμορφώθηκαν επίσης ώστε να μην επαναδημιουργούν σιωπηρά τον κοινό κατάλογο με διαφορετικά δικαιώματα. Αντίστοιχα, το `vdi-status`, το οποίο έχει μόνο εποπτικό ρόλο, ελέγχει την κατάσταση ενός lock χωρίς να δημιουργεί το lock file όταν αυτό απουσιάζει.

Με αυτή τη διαμόρφωση, το per-VM lock αποτελεί κοινό σημείο συγχρονισμού μεταξύ allocation, reset και destruction και όχι ανεξάρτητο μηχανισμό κάθε script.

5.7.2 Καταγραφή και Χειρισμός Σφαλμάτων

Τα scripts υλοποιήθηκαν με αυστηρότερο χειρισμό σφαλμάτων μέσω της βασικής διαμόρφωσης:

```
set -Eeuo pipefail
IFS=$'\n\t'
```

όπου αυτό εφαρμόζεται. Παράλληλα χρησιμοποιούνται έλεγχοι εγκυρότητας πριν από λειτουργίες που μεταβάλλουν την κατάσταση του συστήματος, όπως έλεγχος παραμέτρων, ύπαρξης αρχείων, membership σε pool, περιοχής VMID, tags και snapshots.

Στην περίπτωση του broker απαιτείται επιπλέον διαχωρισμός μεταξύ των διαγνωστικών δεδομένων και της απάντησης που λαμβάνει ο thin client. Το `allocate_vm.sh` διατηρεί το αρχικό standard output σε ξεχωριστό file descriptor και ανακατευθύνει την κανονική έξοδο του script στο:

```
/var/log/vdi-broker/allocate_vm.log
```

Η επιτυχής λειτουργία επιστρέφει στον thin client μόνο την IPv4 διεύθυνση της επιλεγμένης VM, ενώ μία αποτυχία επιστρέφει σύντομο μήνυμα σφάλματος. Με αυτόν τον τρόπο η client-side επεξεργασία δεν επηρεάζεται από τα εσωτερικά diagnostic messages του allocator.

Το `trigger_broker.sh` πραγματοποιεί επιπλέον καταγραφή μέσω `logger` με tag:

```
vdi-trigger
```

ώστε οι ενεργοποιήσεις του broker να είναι διαθέσιμες μέσω του system journal. Αντίστοιχα, ο thin client διατηρεί τοπική καταγραφή της δικής του πλευράς στο:

```
~/vdi-rdp.log
```

Ο συνδυασμός των καταγραφών επιτρέπει τον διαχωρισμό ενός προβλήματος ανάμεσα στην ενεργοποίηση του broker, στην εσωτερική διαδικασία allocation και στη client-side προσπάθεια σύνδεσης.

5.7.3 Τελική Ροή Λειτουργίας

Μετά την ολοκλήρωση των παραπάνω μηχανισμών, η λειτουργική ακολουθία για μία νέα VDI συνεδρία είναι η ακόλουθη:

1. Ο thin client πραγματοποιεί αυτόματη είσοδο στην τοπική κονσόλα και εμφανίζει την επιλογή Linux ή Windows.
2. Η επιλογή αποθηκεύεται στη μεταβλητή `VDI_POOL` και εκκινείται το X environment.
3. Το `vdi-rdp.sh` αποστέλλει μέσω SSH το όνομα του επιλεγμένου pool στον λογαριασμό `lab-launcher` του Proxmox host.
4. Το forced command ενεργοποιεί το `trigger_broker.sh`, το οποίο επικυρώνει την επιλογή και εκτελεί το `allocate_vm.sh` ως `vdi-broker`.
5. Ο allocator διαβάζει το αντίστοιχο pool file και αναζητά stopped VM, χρησιμοποιώντας το κοινό per-VM lock κατά την επιλογή.
6. Η επιλεγμένη VM εκκινείται και ο broker περιμένει την εμφάνιση κατάλληλης IPv4 διεύθυνσης μέσω του QEMU Guest Agent.
7. Η IPv4 διεύθυνση επιστρέφεται ως αποτέλεσμα της SSH σύνδεσης.
8. Ο thin client επικυρώνει την απάντηση και εκκινεί το FreeRDP προς τη συγκεκριμένη VM.
9. Μετά τη χρήση, η VM μπορεί να επανέλθει στο snapshot `rdp-ready` μέσω του lifecycle μηχανισμού, εφόσον ικανοποιούνται οι προβλεπόμενοι έλεγχοι ασφαλούς επαναφοράς.

Η ίδια βασική ροή χρησιμοποιείται και για τα δύο λειτουργικά συστήματα. Οι διαφορές μεταξύ Windows και Linux περιορίζονται κυρίως στην προετοιμασία των guests, στις υπηρεσίες απομακρυσμένης πρόσβασης και σε ορισμένους OS-specific ελέγχους του lifecycle management. Ο thin client και ο broker χρησιμοποιούν κοινή διεπαφή επιλογής pool και allocation.

Παράλληλα, τα lifecycle εργαλεία λειτουργούν πάνω στα ίδια pool files, στα ίδια lifecycle metadata και στον ίδιο μηχανισμό κλειδώματος. Έτσι, η ανάθεση, η επαναφορά, η καταστροφή, η αναπλήρωση και η εποπτεία των disposable VDI VMs αποτελούν επιμέρους λειτουργίες της ίδιας υλοποίησης αντί για ανεξάρτητα scripts χωρίς κοινή κατάσταση.

Η ενότητα αυτή ολοκληρώνει την παρουσίαση της τεχνικής σύνδεσης των υποσυστημάτων. Οι πρακτικές δυσκολίες που εμφανίστηκαν κατά την ανάπτυξη και οι αλλαγές που απαιτήθηκαν για να προκύψει η τελική μορφή της υλοποίησης παρουσιάζονται στην επόμενη ενότητα, ενώ η συστηματική επαλήθευση της συμπεριφοράς του ολοκληρωμένου συστήματος εξετάζεται στο Κεφάλαιο 6.

5.8 Πρακτικά Ζητήματα και Προσαρμογές κατά την Υλοποίηση

Κατά την ανάπτυξη του συστήματος προέκυψαν ορισμένα πρακτικά ζητήματα τα οποία δεν επηρέασαν τη βασική αρχιτεκτονική, αλλά απαιτήσαν αλλαγές στη διαμόρφωση των golden images και στα scripts αυτοματοποίησης. Οι σημαντικότερες από αυτές τις προσαρμογές παρουσιάζονται συνοπτικά στην παρούσα ενότητα, καθώς αποτελούν μέρος της τελικής μορφής της υλοποίησης.

5.8.1 Λήξη Κωδικού του Κοινού Λογαριασμού Windows

Στο Windows περιβάλλον χρησιμοποιήθηκε ο τοπικός λογαριασμός Studin για τις RDP συνδέσεις. Κατά την ανάπτυξη διαπιστώθηκε ότι η πολιτική λήξης του κωδικού πρόσβασης μπορούσε να καταστήσει τον λογαριασμό μη διαθέσιμο για την προβλεπόμενη αυτοματοποιημένη χρήση, ακόμη και όταν η υπόλοιπη διαδικασία allocation και σύνδεσης λειτουργούσε κανονικά. Για τον λόγο αυτό, πριν από την τελική γενίκευση του Windows golden image, η ιδιότητα:

```
PasswordNeverExpires
```

του λογαριασμού Studin ορίστηκε σε ενεργή κατάσταση. Η αλλαγή ενσωματώθηκε στην τελική προετοιμασία του Windows image ώστε τα clones που δημιουργούνται από το template να μην εξαρτώνται από περιοδική χειροκίνητη ανανέωση του κοινού κωδικού.

Η επιλογή αυτή αφορά το συγκεκριμένο πειραματικό μοντέλο κοινών διαπιστευτηρίων. Δεν αποτελεί μηχανισμό εξατομικευμένης ταυτοποίησης χρηστών, αλλά πρακτική προσαρμογή ώστε η λήξη ενός τοπικού κωδικού να μην διακόπτει τη λειτουργία του VDI workflow.

5.8.2 Πρόβλημα Sysprep από AppX Package

Κατά την προετοιμασία του Windows golden image, η διαδικασία Sysprep δεν μπορούσε αρχικά να ολοκληρώσει τη γενίκευση του συστήματος. Η αιτία εντοπίστηκε στο AppX package:

```
MicrosoftWindows.CrossDevice_0.25072.59.0_x64
```

το οποίο είχε εγκατασταθεί σε κατάσταση που δεν ήταν συμβατή με τη διαδικασία γενίκευσης. Τα Sysprep logs συσχέτισαν το συγκεκριμένο package με σφάλμα 0x80073cf2.

Η διόρθωση πραγματοποιήθηκε με αφαίρεση του package από το user context και από την εγκατάσταση για όλους τους χρήστες, όπου αυτό ήταν καταχωρισμένο. Μετά την αφαίρεσή του, το Windows image μπορούσε να χρησιμοποιηθεί για την προβλεπόμενη διαδικασία `/generalize /oobe /shutdown`.

Η προσαρμογή αυτή πραγματοποιήθηκε στο mutable golden image πριν από τη δημιουργία του τελικού template, ώστε το συγκεκριμένο πρόβλημα να μην μεταφερθεί στα VDI clones.

5.8.3 Πρώτη Εκκίνηση των Windows Clones

Η συμπεριφορά των Windows clones μετά το Sysprep διαφοροποιείται από αυτή των Linux clones. Ένα νέο Windows clone πρέπει να ολοκληρώσει την πρώτη post-Sysprep εκκίνηση και τα σχετικά βήματα αρχικοποίησης πριν θεωρηθεί κατάλληλο για τη δημιουργία του καθαρού lifecycle snapshot.

Για την αυτοματοποίηση αυτής της μετάβασης χρησιμοποιήθηκε το:

```
C:\Windows\Setup\Scripts\SetupComplete.cmd
```

με εντολή αυτόματου τερματισμού:

```
shutdown /s /t 0 /f
```

Το `vdi-create-pool` προσαρμόστηκε αντίστοιχα ώστε, για Windows clones, να πραγματοποιεί την πρώτη εκκίνηση και να περιμένει την επιστροφή της VM σε stopped κατάσταση. Μόνο τότε συνεχίζει με τα lifecycle metadata, το snapshot `rdp-ready` και την καταχώριση στο pool.

Με αυτόν τον τρόπο η διαφορετική διαδικασία αρχικοποίησης των Windows ενσωματώθηκε στο κοινό εργαλείο δημιουργίας pool χωρίς να απαιτείται χειροκίνητη προετοιμασία κάθε clone.

5.8.4 Σειρά Εφαρμογής Tags και Δημιουργίας Snapshot

Κατά την ανάπτυξη του lifecycle μηχανισμού έγινε εμφανές ότι η σειρά με την οποία εφαρμόζονται τα lifecycle tags και δημιουργείται το snapshot έχει λειτουργική σημασία. Η επαναφορά ενός Proxmox snapshot δεν αφορά μόνο την κατάσταση των virtual disks, αλλά μπορεί να επαναφέρει και στοιχεία της διαμόρφωσης της VM.

Εάν το snapshot `rdp-ready` δημιουργηθεί πριν από την εφαρμογή των τελικών lifecycle tags, μία μεταγενέστερη επαναφορά μπορεί να επαναφέρει και την παλαιότερη κατάσταση των tags. Αυτό δημιουργεί ασυνέπεια, επειδή τα scripts χρησιμοποιούν τα tags ως μέρος των ελέγχων για το αν μία VM αποτελεί διαχειριζόμενο και έτοιμο VDI μέλος.

Για τον λόγο αυτό η τελική σειρά στο `vdi-create-pool` καθορίστηκε ως:

```

apply lifecycle tags
->
create rdp-ready snapshot
->
register VM in pool file

```

Έτσι, η κατάσταση που αποθηκεύεται στο `rdp-ready snapshot` περιλαμβάνει ήδη τα `metadata` που αναμένουν τα υπόλοιπα `lifecycle` εργαλεία.

5.8.5 Ανίχνευση Ενεργών RDP Συνεδριών

Η ασφαλής λειτουργία του `vdi-reset-pool` απαιτεί διάκριση μεταξύ μιας `running VM` που δεν χρησιμοποιείται πλέον και μιας `VM` στην οποία υπάρχει ενεργή απομακρυσμένη συνεδρία. Στα Windows χρησιμοποιήθηκε συνδυασμός του `quser` και ελέγχου των TCP συνδέσεων στη θύρα 3389. Η αρχική επεξεργασία της PowerShell εξόδου μπορούσε να οδηγήσει σε αμφίσημα αποτελέσματα, επειδή η ύπαρξη οποιασδήποτε `textual` εξόδου δεν αντιστοιχεί κατ' ανάγκη σε ενεργή RDP σύνδεση.

Για τον λόγο αυτό ο PowerShell έλεγχος προσαρμόστηκε ώστε να παράγει έναν συγκεκριμένο δείκτη μόνο όταν εντοπίζεται `established` σύνδεση:

```
ACTIVE_RDP
```

Το `lifecycle script` αναζητά πλέον αυτόν τον δείκτη αντί να θεωρεί οποιαδήποτε μη κενή έξοδο ως απόδειξη ενεργής συνεδρίας.

Στα Linux guests η ανίχνευση βασίστηκε στην έξοδο του:

```
ss -tnp
```

με αναζήτηση `established` σύνδεσης στην τοπική θύρα 3389 του XRDP. Οι δύο OS-specific έλεγχοι ενσωματώθηκαν στο ίδιο `reset workflow`, διατηρώντας διαφορετικό μηχανισμό ανίχνευσης όπου αυτό απαιτείται από το guest λειτουργικό σύστημα.

5.8.6 Κοινός Χώρος Locks και Δικαιώματα Πρόσβασης

Η χρήση `per-VM locks` είχε αρχικά εφαρμοστεί ως μηχανισμός αποφυγής ταυτόχρονων μεταβολών της ίδιας `VM`. Ωστόσο, η τελική ολοκλήρωση του `allocator` με τα `lifecycle scripts` απαίτησε τα ίδια `lock files` να μπορούν να χρησιμοποιηθούν από διεργασίες που εκτελούνται με διαφορετικές ταυτότητες.

Ο `allocator` εκτελείται ως `vdi-broker`, ενώ τα διαχειριστικά `lifecycle` εργαλεία μπορούν να εκτελούνται ως `root`. Επομένως, `lock files` που δημιουργούνταν αποκλειστικά με `ownership` και `permissions` του `root` δεν μπορούσαν να αποτελέσουν πραγματικά κοινό μηχανισμό συγχρονισμού.

Η τελική προσαρμογή ήταν η χρήση κοινής ομάδας `vdi-broker`, `set-GID runtime directory` και `group-writable lock files`, σε συνδυασμό με κατάλληλο `umask`. Προστέθηκε επίσης κανόνας `systemd-tmpfiles` ώστε ο `runtime κατάλογος` να επαναδημιουργείται με τα σωστά δικαιώματα μετά από `reboot`. Η τελική διαμόρφωση παρουσιάστηκε αναλυτικά στην προηγούμενη ενότητα.

Η αλλαγή αυτή μετέτρεψε τα locks από τοπική σύμβαση των επιμέρους scripts σε κοινό μηχανισμό συντονισμού μεταξύ allocation, reset και destruction.

5.8.7 Αυτοματοποίηση της Εκκίνησης του Thin Client

Κατά τα πρώτα στάδια ανάπτυξης, η έναρξη του γραφικού περιβάλλοντος του thin client πραγματοποιούνταν χειροκίνητα με `startx`. Η προσέγγιση αυτή ήταν κατάλληλη για την αρχική ανάπτυξη και τον έλεγχο των επιμέρους components, αλλά δεν αντιστοιχούσε στην επιθυμητή τελική εμπειρία χρήσης.

Η τελική διαμόρφωση μετακίνησε τη διαδικασία σε αυτοματοποιημένη ακολουθία:

```
boot
-> tty1 autologin
-> OS selection menu
-> export VDI_POOL
-> startx
-> .xinitrc
-> vdi-rdp.sh
```

Η μεταβλητή `VDI_POOL` χρησιμοποιήθηκε για να μεταφερθεί η επιλογή Linux ή Windows από το console menu στη γραφική συνεδρία και τελικά στο αίτημα προς τον broker. Έτσι αφαιρέθηκε η ανάγκη χειροκίνητης εκτέλεσης του X environment και το thin-client workflow περιορίστηκε στην επιλογή του επιθυμητού λειτουργικού συστήματος.

Οι παραπάνω προσαρμογές δεν άλλαξαν τη βασική αρχιτεκτονική του συστήματος, αλλά διαμόρφωσαν σημαντικά την τελική υλοποίησή του. Αντιμετώπισαν προβλήματα που σχετίζονταν με την προετοιμασία των Windows images, τη διατήρηση συνεπούς lifecycle state, την ασφαλή αναγνώριση ενεργών συνεδριών, τον συγχρονισμό των scripts και την αυτοματοποίηση της πλευράς του thin client.

5.9 Σύνοψη Κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκε η πρακτική υλοποίηση του προτεινόμενου συστήματος VDI στο πειραματικό περιβάλλον Proxmox VE. Προετοιμάστηκαν ξεχωριστά Windows και Linux golden images και δημιουργήθηκαν τα αντίστοιχα templates, από τα οποία παράγονται linked clones για τη συγκρότηση των δύο VDI pools. Παράλληλα, υλοποιήθηκε ο thin client και ο broker που συνδέουν την επιλογή του χρήστη με την αυτόματη ανάθεση, εκκίνηση και απομακρυσμένη πρόσβαση σε διαθέσιμη εικονική μηχανή.

Η βασική διαδικασία allocation συμπληρώθηκε από μηχανισμούς διαχείρισης του κύκλου ζωής των disposable VDI πόρων. Τα εργαλεία που υλοποιήθηκαν επιτρέπουν τη δημιουργία και αναπλήρωση των pools, την επαναφορά των εικονικών μηχανών στην προκαθορισμένη κατάσταση `rdp-ready`, την ελεγχόμενη καταστροφή μεμονωμένων ή όλων των μελών ενός pool και τη συγκεντρωτική εποπτεία της κατάστασής τους. Η λειτουργία των επιμέρους εργαλείων ενοποιήθηκε μέσω κοινών αρχείων διαμόρφωσης και pool membership, lifecycle metadata και κοινού μηχανισμού per-VM locking.

Κατά την ανάπτυξη απαιτήθηκαν επιπλέον προσαρμογές στην προετοιμασία των Windows images, στην ανίχνευση ενεργών RDP συνεδριών, στη σειρά εφαρμογής των lifecycle metadata, στα δικαιώματα του κοινού μηχανισμού locking και στην αυτοματοποίηση του thin client. Με την ενσωμάτωση των αλλαγών αυτών διαμορφώθηκε η τελική υλοποίηση που αποτελεί τη βάση για τις δοκιμές του συστήματος. Στο επόμενο κεφάλαιο παρουσιάζεται η συστηματική επαλήθευση των επιμέρους λειτουργιών και της ολοκληρωμένης ροής, καθώς και τα αποτελέσματα και οι περιορισμοί που προέκυψαν από την αξιολόγησή τους.

Κεφάλαιο 6

Δοκιμές, Επαλήθευση και Αξιολόγηση του Συστήματος

6.1 Εισαγωγή

Μετά την ολοκλήρωση της υλοποίησης, πραγματοποιήθηκε σειρά λειτουργικών ελέγχων με σκοπό την επαλήθευση της συμπεριφοράς του συστήματος VDI υπό τις βασικές συνθήκες λειτουργίας για τις οποίες σχεδιάστηκε. Οι έλεγχοι επικεντρώθηκαν στο τελικό, ολοκληρωμένο σύστημα και όχι στα επιμέρους στάδια ανάπτυξης που προηγήθηκαν.

Η αξιολόγηση καλύπτει τη δημιουργία και διατήρηση των pools εικονικών μηχανών, την ανάθεση διαθέσιμων VMs μέσω του broker, την απομακρυσμένη πρόσβαση σε περιβάλλοντα Windows και Linux, την προστασία ενεργών συνεδριών κατά τη διαδικασία επαναφοράς, την επαναφορά των VMs σε καθαρή κατάσταση, καθώς και τις λειτουργίες καταστροφής και αναπλήρωσης των διαχειριζόμενων πόρων. Επιπλέον, εξετάστηκε η συμπεριφορά του συστήματος κατά την ταυτόχρονη υποβολή αιτημάτων, ώστε να επαληθευτεί η ορθή λειτουργία του μηχανισμού κλειδώματος.

Για κάθε έλεγχο παρουσιάζονται οι συνθήκες εκτέλεσης, το παρατηρούμενο αποτέλεσμα και η αντίστοιχη αξιολόγηση. Δεν επαναλαμβάνεται η αναλυτική περιγραφή των μηχανισμών που υλοποιήθηκαν, καθώς αυτή έχει ήδη παρουσιαστεί στα προηγούμενα κεφάλαια. Όπου κρίνεται χρήσιμο, παρατίθενται επιλεγμένα αποτελέσματα εντολών ως τεκμήρια της παρατηρούμενης συμπεριφοράς.

Οι δοκιμές πραγματοποιήθηκαν στο πειραματικό περιβάλλον Proxmox VE που χρησιμοποιήθηκε κατά την ανάπτυξη του συστήματος, το οποίο λειτουργούσε σε εικονικοποιημένη πειραματική υποδομή. Συνεπώς, τα αποτελέσματα τεκμηριώνουν τη λειτουργική ορθότητα της συγκεκριμένης υλοποίησης και δεν αποτελούν μετρήσεις απόδοσης ή κλιμάκωσης σε παραγωγικό περιβάλλον μεγάλης κλίμακας.

6.2 Έλεγχος Δημιουργίας και Κατάστασης των VDI Pools

Ο πρώτος έλεγχος αφορούσε την ορθή δημιουργία και καταχώριση των διαχειριζόμενων εικονικών μηχανών στα δύο pools του συστήματος. Στην τελική διαμόρφωση είχε οριστεί επιθυμητό μέγεθος δύο εικονικών μηχανών για το Linux pool και δύο για το Windows pool.

Η διαδικασία δημιουργίας εκτελέστηκε αρχικά σε κατάσταση κατά την οποία τα αντίστοιχα pools δεν διέθεταν τον απαιτούμενο αριθμό έγκυρων μελών. Το σύστημα αναγνώρισε το έλλειμμα και δημιούργησε μόνο τον αριθμό των εικονικών μηχανών που απαιτούνταν για την επίτευξη του προκαθορισμένου μεγέθους. Για το Linux pool χρησιμοποιήθηκαν τα VMIDs 120 και 121, ενώ για το Windows pool τα VMIDs 130 και 131.

Μετά την ολοκλήρωση της διαδικασίας, ελέγχθηκε η κατάσταση των τεσσάρων VMs. Και στις δύο περιπτώσεις οι εικονικές μηχανές είχαν καταχωριστεί στα αντίστοιχα pool files, διέθεταν το snapshot `rdp-ready` και έφεραν τα αναμενόμενα lifecycle tags. Η τελική κατάσταση των pools συνοψίζεται στον Πίνακα 6.1.

Πίνακας 6.1: Τελική κατάσταση των VDI pools μετά τη δημιουργία

Pool	VMID	Κατάσταση	Snapshot	Tags
Linux	120	stopped	ready	OK
Linux	121	stopped	ready	OK
Windows	130	stopped	ready	OK
Windows	131	stopped	ready	OK

Επιπλέον, επαναλαμβανόμενος έλεγχος της διαδικασίας δημιουργίας σε pool που είχε ήδη φτάσει στο προκαθορισμένο μέγεθός του δεν οδήγησε στη δημιουργία επιπλέον VMs. Με τον τρόπο αυτό επιβεβαιώθηκε ότι η διαδικασία διατηρεί το επιθυμητό μέγεθος του pool χωρίς να δημιουργεί πλεονάζοντες πόρους.

Τα αποτελέσματα του ελέγχου επιβεβαίωσαν ότι η αυτοματοποιημένη διαδικασία μπορούσε να αποκαταστήσει την προκαθορισμένη χωρητικότητα των δύο pools και ότι τα μέλη που καταχωρίστηκαν μετά την ολοκλήρωσή της διέθεταν την αναμενόμενη κατάσταση, το snapshot επαναφοράς και τα απαιτούμενα lifecycle tags.

6.3 Έλεγχος Ανάθεσης και Απομακρυσμένης Πρόσβασης

Μετά την επιβεβαίωση της ορθής κατάστασης των pools, πραγματοποιήθηκαν έλεγχοι ανάθεσης εικονικών μηχανών μέσω του broker και σύνδεσης από τον thin client. Οι δοκιμές πραγματοποιήθηκαν ανεξάρτητα για τα δύο υποστηριζόμενα λειτουργικά συστήματα, ώστε να επιβεβαιωθεί η πλήρης διαδρομή από την επιλογή του χρήστη μέχρι την εμφάνιση του απομακρυσμένου περιβάλλοντος εργασίας.

6.3.1 Έλεγχος Windows Pool

Κατά τον έλεγχο του Windows pool επιλέχθηκε η αντίστοιχη επιλογή από το μενού του thin client και υποβλήθηκε αίτημα ανάθεσης στον broker. Ο broker επέλεξε διαθέσιμη εικονική μηχανή του Windows pool, την εκκίνησε και επέστρεψε τη διεύθυνση IPv4 που ανακτήθηκε μετά την εκκίνησή της.

Στη συνέχεια πραγματοποιήθηκε σύνδεση μέσω RDP στη διεύθυνση που επιστράφηκε. Η σύνδεση ολοκληρώθηκε επιτυχώς και εμφανίστηκε το περιβάλλον εργασίας της Windows εικονικής μηχανής με τον λογαριασμό `Studin`. Με τον τρόπο αυτό επαληθεύτηκε η λειτουργία της πλήρους διαδρομής `thin client--broker--Windows VM`.

Κατά τη διάρκεια των δοκιμών επιβεβαιώθηκε επίσης η χρήση περισσότερων του ενός μελών του pool. Σε περίπτωση κατά την οποία μία εικονική μηχανή δεν ήταν διαθέσιμη για νέα σύνδεση λόγω υφιστάμενης συνεδρίας, η διαδικασία συνέχισε στην επόμενη διαθέσιμη VM και η σύνδεση ολοκληρώθηκε επιτυχώς.

6.3.2 Έλεγχος Linux Pool

Αντίστοιχη διαδικασία ακολουθήθηκε για το Linux pool. Μετά την επιλογή Linux από το μενού του thin client, ο broker επέλεξε διαθέσιμο μέλος του αντίστοιχου pool και επέστρεψε την IPv4 διεύθυνσή του μετά την εκκίνησή του.

Η σύνδεση μέσω XRDP ολοκληρώθηκε επιτυχώς και ο thin client απέκτησε πρόσβαση στο γραφικό περιβάλλον XFCE της Linux εικονικής μηχανής. Η επιτυχής σύνδεση επιβεβαίωσε ότι η ίδια διαδικασία ανάθεσης και απομακρυσμένης πρόσβασης μπορούσε να χρησιμοποιηθεί και για το δεύτερο υποστηριζόμενο λειτουργικό σύστημα.

6.3.3 Αξιολόγηση της End-to-End Ροής

Τα αποτελέσματα των δύο δοκιμών έδειξαν ότι η τελική ροή λειτουργίας ολοκληρωνόταν χωρίς να απαιτείται από τον χρήστη γνώση του VMID ή της IP διεύθυνσης της εικονικής μηχανής. Η μόνη απαιτούμενη επιλογή στον thin client ήταν το επιθυμητό λειτουργικό σύστημα, ενώ η επιλογή του συγκεκριμένου VM και η επιστροφή της διεύθυνσης σύνδεσης πραγματοποιήθηκαν αυτόματα.

Συνεπώς, η end-to-end λειτουργία επαληθεύτηκε τόσο για το Windows όσο και για το Linux pool. Οι δοκιμές αυτές αποτελούν λειτουργική επαλήθευση της διαδικασίας πρόσβασης στο συγκεκριμένο πειραματικό περιβάλλον και δεν χρησιμοποιούνται ως μέτρηση της απόδοσης ή της ποιότητας της σύνδεσης RDP/XRDP.

6.4 Έλεγχος Ταυτόχρονων Αιτημάτων Ανάθεσης

Για την επαλήθευση της συμπεριφοράς του συστήματος σε περίπτωση ταυτόχρονων αιτημάτων, πραγματοποιήθηκε δοκιμή με δύο παράλληλες κλήσεις προς τον allocator του Linux pool. Ως αρχική συνθήκη, οι δύο εικονικές μηχανές του pool, με VMIDs 120 και 121, βρίσκονταν σε κατάσταση stopped και ήταν διαθέσιμες για ανάθεση.

Κατά την πρώτη εκτέλεση της δοκιμής, και τα δύο αιτήματα ολοκληρώθηκαν ανεπιτυχώς με το μήνυμα:

```
Error: No available VDI machines at this time.
```

Το αποτέλεσμα δεν αντιστοιχούσε στην αναμενόμενη λειτουργία, καθώς υπήρχαν δύο διαθέσιμα μέλη στο pool. Ο έλεγχος που ακολούθησε έδειξε ότι τα κοινά αρχεία κλειδώματος είχαν δημιουργηθεί με δικαιώματα που δεν επέτρεπαν στον λογαριασμό vdi-broker να τα ανοίξει για εγγραφή. Η άμεση δοκιμή πρόσβασης σε αρχείο κλειδώματος από τον συγκεκριμένο λογαριασμό επέστρεψε:

```
Permission denied
```

Μετά τον εντοπισμό του προβλήματος διορθώθηκαν τα δικαιώματα του κοινού χώρου κλειδώματος και η αλλαγή ενσωματώθηκε στην τελική διαμόρφωση του συστήματος. Πριν από την επανάληψη του ελέγχου επιβεβαιώθηκε ότι ο λογαριασμός `vdi-broker` μπορούσε πλέον να αποκτήσει το αντίστοιχο lock.

Η ίδια δοκιμή επαναλήφθηκε στη συνέχεια με δύο παράλληλα αιτήματα και τις δύο Linux VMs διαθέσιμες. Αυτή τη φορά και οι δύο διεργασίες ολοκληρώθηκαν επιτυχώς:

```
=== REQUEST 1 ===  
Exit code: 0  
192.168.6.125
```

```
=== REQUEST 2 ===  
Exit code: 0  
192.168.6.124
```

Μετά την ολοκλήρωση των δύο αιτημάτων, οι VMIDs 120 και 121 βρίσκονταν ταυτόχρονα σε κατάσταση `running`. Επομένως, τα δύο παράλληλα αιτήματα εξυπηρετήθηκαν από διαφορετικές εικονικές μηχανές και δεν παρατηρήθηκε διπλή ανάθεση του ίδιου διαθέσιμου μέλους. Η δοκιμή αυτή είχε ιδιαίτερη σημασία, καθώς ο αρχικός έλεγχος αποκάλυψε πρόβλημα δικαιωμάτων το οποίο δεν είχε εμφανιστεί στις προηγούμενες σειριακές δοκιμές. Μετά τη διόρθωση και την επανάληψη του ίδιου ελέγχου, επιβεβαιώθηκε η ορθή λειτουργία του μηχανισμού συγχρονισμού για το εξεταζόμενο σενάριο δύο ταυτόχρονων αιτημάτων.

6.5 Έλεγχος Επαναφοράς και Προστασίας Ενεργών Συνεδριών

Η διαδικασία επαναφοράς ελέγχθηκε υπό διαφορετικές καταστάσεις λειτουργίας των εικονικών μηχανών, ώστε να διαπιστωθεί ότι η επαναφορά πραγματοποιείται μόνο όταν δεν υπάρχει κίνδυνος διακοπής μιας ενεργής συνεδρίας. Οι δοκιμές περιέλαβαν τόσο Windows όσο και Linux VMs.

6.5.1 Έλεγχος Περιόδου Προστασίας

Αρχικά εξετάστηκε η συμπεριφορά σε VM που είχε εκκινηθεί πρόσφατα. Η εικονική μηχανή βρισκόταν σε κατάσταση `running`, αλλά ο χρόνος λειτουργίας της δεν είχε υπερβεί το καθορισμένο όριο των 300 s.

Κατά την εκτέλεση της διαδικασίας επαναφοράς, η συγκεκριμένη VM παραλείφθηκε και δεν πραγματοποιήθηκε rollback. Μετά την παρέλευση του ορίου, η ίδια VM μπορούσε να εξεταστεί κανονικά ως υποψήφια για επαναφορά.

Το αποτέλεσμα επιβεβαίωσε ότι μια πρόσφατα εκκινημένη εικονική μηχανή δεν επαναφέρεται πρόωρα κατά το χρονικό διάστημα στο οποίο ενδέχεται να βρίσκεται ακόμη σε διαδικασία εκκίνησης ή εγκαθίδρυσης σύνδεσης.

6.5.2 Έλεγχος Ενεργής Συνεδρίας

Ο επόμενος έλεγχος πραγματοποιήθηκε ενώ υπήρχε ενεργή απομακρυσμένη συνεδρία. Στην περίπτωση Windows VM, η διαδικασία εντόπισε την ενεργή σύνδεση RDP και παρέλειψε την επαναφορά:

```
SKIP: active Windows RDP session detected.
```

Η εικονική μηχανή παρέμεινε σε λειτουργία και η ενεργή συνεδρία δεν διακόπηκε. Μετά την αποσύνδεση του χρήστη επαναλήφθηκε ο έλεγχος και δεν εντοπίστηκε πλέον ενεργή συνεδρία. Αντίστοιχη δοκιμή πραγματοποιήθηκε και σε Linux VM με ενεργή σύνδεση XRDP. Όσο η σύνδεση παρέμενε ενεργή, η VM εξαιρέθηκε από τη διαδικασία επαναφοράς. Μετά τον τερματισμό της συνεδρίας, η σύνδεση δεν ανιχνευόταν πλέον και η εικονική μηχανή μπορούσε να προχωρήσει στο επόμενο στάδιο του ελέγχου.

Τα αποτελέσματα των δύο περιπτώσεων επιβεβαίωσαν ότι η παρουσία ενεργής απομακρυσμένης συνεδρίας απέτρεπε την επαναφορά τόσο στο Windows όσο και στο Linux pool.

6.5.3 Rollback και Επαναχρησιμοποίηση

Μετά την απουσία ενεργής συνεδρίας και εφόσον είχε παρέλθει η περίοδος προστασίας, πραγματοποιήθηκε επαναφορά της εξεταζόμενης VM στο snapshot `rdp-ready`. Η διαδικασία ολοκληρώθηκε επιτυχώς και η VM επέστρεψε σε κατάσταση `stopped`.

Για την επιβεβαίωση ότι η επαναφορά δεν άφησε την εικονική μηχανή σε μη χρησιμοποιήσιμη κατάσταση, πραγματοποιήθηκε νέο αίτημα ανάθεσης. Η VM εκκινήθηκε ξανά και πραγματοποιήθηκε επιτυχής απομακρυσμένη σύνδεση. Με αυτόν τον τρόπο επαληθεύτηκε ο πλήρης κύκλος χρήσης, επαναφοράς και επαναχρησιμοποίησης της ίδιας εικονικής μηχανής.

Κατά τις δοκιμές εντοπίστηκε επίσης ασυνέπεια στη διατήρηση των lifecycle tags μετά από rollback. Το πρόβλημα διορθώθηκε στην τελική διαδικασία προετοιμασίας των VMs και, μετά τη δημιουργία νέου snapshot, η επαναφορά διατηρούσε την αναμενόμενη κατάσταση των tags. Η διορθωμένη συμπεριφορά επιβεβαιώθηκε στις επόμενες δοκιμές.

6.5.4 Έλεγχος Επαναφοράς Ολόκληρου Pool

Τέλος, η διαδικασία εφαρμόστηκε στο σύνολο του Linux pool. Οι VMIDs 120 και 121 βρίσκονταν σε κατάσταση `running`, είχαν υπερβεί το όριο των 300 s και δεν παρουσίαζαν ενεργές συνεδρίες XRDP.

Και οι δύο εικονικές μηχανές πέρασαν επιτυχώς τον τελικό έλεγχο ενεργής συνεδρίας, σταμάτησαν και επανήλθαν στο snapshot `rdp-ready`. Ο μεταγενέστερος έλεγχος κατάστασης έδειξε:

```
120  linux-vdi-01  stopped  -  n/a  ready  OK  free
121  linux-vdi-02  stopped  -  n/a  ready  OK  free
```

Η δοκιμή επιβεβαίωσε ότι η διαδικασία μπορούσε να εφαρμοστεί σε όλα τα επιλέξιμα μέλη ενός pool, διατηρώντας παράλληλα τους ελέγχους προστασίας ανεξάρτητα για κάθε VM.

6.6 Έλεγχος Καταστροφής και Αναπλήρωσης του Pool

Για την επαλήθευση της δυνατότητας ελεγχόμενης αφαίρεσης διαχειριζόμενων εικονικών μηχανών και της μετέπειτα αποκατάστασης του pool, πραγματοποιήθηκε δοκιμή στο Linux pool. Πριν από την έναρξη του ελέγχου, το pool περιείχε τις VMIDs 120 και 121 και βρισκόταν στην επιθυμητή χωρητικότητα των δύο VMs.

6.6.1 Καταστροφή Ολόκληρου του Pool

Η διαδικασία καταστροφής εκτελέστηκε για ολόκληρο το Linux pool. Και οι δύο εικονικές μηχανές βρίσκονταν σε κατάσταση `stopped` και πέρασαν επιτυχώς τους προβλεπόμενους ελέγχους πριν από την αφαίρεσή τους.

Η καταστροφή ολοκληρώθηκε επιτυχώς και για τις δύο VMs:

```
=== VM 120 ===
Current status: stopped
SUCCESS: managed VDI VM 120 was destroyed.
```

```
=== VM 121 ===
Current status: stopped
SUCCESS: managed VDI VM 121 was destroyed.
```

Destruction operation complete.

Μετά την ολοκλήρωση της διαδικασίας, το αρχείο του Linux pool ήταν κενό. Επιπλέον, ανεξάρτητος έλεγχος μέσω του Proxmox επιβεβαίωσε ότι οι VMIDs 120 και 121 δεν υπήρχαν πλέον. Η κατάσταση του pool αποτυπώθηκε ως:

```
Linux pool
Registered: 0 / Target: 2
```

Το Windows pool παρέμεινε ανεπηρέαστο και συνέχισε να περιέχει τα δύο καταχωρισμένα μέλη του. Το αποτέλεσμα επιβεβαίωσε ότι η λειτουργία καταστροφής περιορίστηκε στο επιλεγμένο pool.

6.6.2 Αυτόματη Αναπλήρωση της Χωρητικότητας

Στη συνέχεια εκτελέστηκε η διαδικασία δημιουργίας για το κενό Linux pool. Κατά τον αρχικό έλεγχο καταγράφηκε η ακόλουθη κατάσταση:

```
Valid pool size:    0
Required size:      2
VMs to create:      2
```

Το σύστημα επέλεξε εκ νέου τις ελεύθερες VMIDs 120 και 121 και δημιούργησε δύο νέα μέλη από το Linux template. Μετά την ολοκλήρωση της προετοιμασίας τους, οι δύο VMs καταχωρίστηκαν εκ νέου στο `linux.pool`.

Ο τελικός έλεγχος κατάστασης έδειξε:

```
Linux pool
```

```
Registered: 2 / Target: 2
```

```
120 linux-vdi-01 stopped - n/a ready OK free
121 linux-vdi-02 stopped - n/a ready OK free
```

Επομένως, το pool επανήλθε από κατάσταση πλήρους απουσίας διαχειριζόμενων VMs στην προκαθορισμένη χωρητικότητα των δύο μελών. Οι νέες εικονικές μηχανές διέθεταν το αναμενόμενο snapshot, τα απαιτούμενα lifecycle tags και ελεύθερη κατάσταση κλειδώματος.

6.6.3 Αξιολόγηση του Κύκλου Καταστροφής και Αναπλήρωσης

Η συνολική μεταβολή της κατάστασης του Linux pool κατά τη δοκιμή ήταν:

$$2/2 \longrightarrow 0/2 \longrightarrow 2/2$$

Το αποτέλεσμα επαλήθευσε ότι η αφαίρεση των διαχειριζόμενων VMs δεν αφήνει μη έγκυρες καταχωρίσεις στο pool και ότι η διαδικασία δημιουργίας μπορεί στη συνέχεια να αναγνωρίσει το έλλειμμα και να επαναφέρει την προκαθορισμένη χωρητικότητα.

Κατά τη δημιουργία των νέων linked clones εμφανίστηκαν επίσης προειδοποιήσεις του LVM-thin σχετικά με τη διαθέσιμη φυσική χωρητικότητα. Οι προειδοποιήσεις δεν εμπόδισαν την ολοκλήρωση της συγκεκριμένης δοκιμής και αξιολογούνται συνολικά στην ενότητα των περιορισμών της πειραματικής επαλήθευσης.

6.7 Έλεγχος Παρακολούθησης της Κατάστασης

Η λειτουργία παρακολούθησης χρησιμοποιήθηκε σε διαφορετικά στάδια των δοκιμών για την επιβεβαίωση της κατάστασης των δύο VDI pools. Μέσω της εντολής `vdi-status` ελέγχθηκαν ο αριθμός των καταχωρισμένων VMs σε σχέση με την επιθυμητή χωρητικότητα, η κατάσταση λειτουργίας κάθε VM, η ύπαρξη του snapshot επαναφοράς, η εγκυρότητα των lifecycle tags και η κατάσταση των αντίστοιχων locks.

Οι εμφανιζόμενες πληροφορίες ακολούθησαν με συνέπεια τις μεταβολές που προκαλούσαν οι προηγούμενοι έλεγχοι. Μετά την παράλληλη ανάθεση, οι δύο Linux VMs εμφανίστηκαν σε κατάσταση `running`, μετά την επαναφορά τους σε κατάσταση `stopped`, ενώ κατά την καταστροφή και αναπλήρωση του Linux pool η καταγεγραμμένη χωρητικότητα μεταβλήθηκε διαδοχικά από 2/2 σε 0/2 και ξανά σε 2/2.

Κατά τον τελικό έλεγχο, τόσο το Linux όσο και το Windows pool βρίσκονταν στην επιθυμητή χωρητικότητα των δύο VMs, με τα καταχωρισμένα μέλη σε κατάσταση `stopped`, έγκυρο snapshot, ορθά tags και ελεύθερα locks. Οι παρατηρήσεις αυτές επιβεβαίωσαν ότι η λειτουργία παρακολούθησης παρείχε συνεπή εικόνα της πραγματικής κατάστασης των διαχειριζόμενων πόρων κατά τη διάρκεια των πειραματικών ελέγχων.

6.8 Αξιολόγηση των Αποτελεσμάτων

6.8.1 Λειτουργική Κάλυψη της Υλοποίησης

Οι δοκιμές κάλυψαν τα βασικά στάδια του κύκλου ζωής που υποστηρίζει η τελική υλοποίηση, από τη δημιουργία και ανάθεση των εικονικών μηχανών έως την επαναφορά, επαναχρησιμοποίηση, καταστροφή και αναπλήρωσή τους. Η επιτυχής πρόσβαση σε περιβάλλοντα Windows και Linux επιβεβαίωσε επίσης τη λειτουργική ολοκλήρωση της διαδρομής από τον thin client έως την επιλεγμένη εικονική μηχανή.

Ιδιαίτερα η μετάβαση του Linux pool από την κανονική κατάσταση 2/2 σε 0/2 και στη συνέχεια ξανά σε 2/2 έδειξε ότι οι επιμέρους lifecycle λειτουργίες μπορούσαν να συνδυαστούν σε έναν πλήρη κύκλο διαχείρισης. Σε συνδυασμό με τις δοκιμές επαναφοράς και επαναχρησιμοποίησης, τα αποτελέσματα τεκμηριώνουν τη λειτουργική συνοχή της υλοποίησης στο εξεταζόμενο περιβάλλον.

6.8.2 Αξιολόγηση των Μηχανισμών Προστασίας

Οι έλεγχοι δεν περιορίστηκαν στις αναμενόμενες επιτυχείς διαδρομές. Εξετάστηκαν επίσης καταστάσεις στις οποίες μια ενέργεια έπρεπε να αναβληθεί ή να αποτραπεί. VM που δεν είχε υπερβεί την περίοδο προστασίας δεν επαναφέρθηκε, ενώ η παρουσία ενεργής συνεδρίας RDP ή XRDP απέτρεψε την εκτέλεση rollback. Μετά την αποσύνδεση του χρήστη, η ίδια διαδικασία μπορούσε να συνεχιστεί κανονικά.

Η δοκιμή παράλληλων αιτημάτων είχε επιπλέον ως αποτέλεσμα τον εντοπισμό προβλήματος στα δικαιώματα του κοινού χώρου κλειδώματος. Το πρόβλημα διορθώθηκε στην τελική διαμόρφωση και η ίδια δοκιμή επαναλήφθηκε επιτυχώς, με τα δύο ταυτόχρονα αιτήματα να εξυπηρετούνται από διαφορετικές εικονικές μηχανές. Το αποτέλεσμα αυτό αποτελεί ταυτόχρονα επαλήθευση του διορθωμένου μηχανισμού και παράδειγμα της χρησιμότητας της πειραματικής διαδικασίας για τον εντοπισμό σφαλμάτων που δεν είχαν εμφανιστεί στις προηγούμενες σειριακές δοκιμές.

Συνολικά, οι παραπάνω παρατηρήσεις δείχνουν ότι οι μηχανισμοί προστασίας που εξετάστηκαν δεν λειτουργούν μόνο ως έλεγχοι εγκυρότητας της κανονικής ροής, αλλά αποτρέπουν συγκεκριμένες ανεπιθύμητες ενέργειες στις καταστάσεις που δοκιμάστηκαν.

6.8.3 Όρια της Πειραματικής Επαλήθευσης

Παρότι οι λειτουργικές δοκιμές κάλυψαν τα βασικά στάδια του κύκλου ζωής της τελικής υλοποίησης, τα αποτελέσματά τους πρέπει να εξεταστούν εντός των ορίων του πειραματικού περιβάλλοντος στο οποίο πραγματοποιήθηκαν.

Η υποδομή αναπτύχθηκε και δοκιμάστηκε σε περιβάλλον Proxmox VE το οποίο εκτελούνταν ως εικονική μηχανή στο πλαίσιο της διαθέσιμης πειραματικής υποδομής. Κατά συνέπεια, η αξιολόγηση πραγματοποιήθηκε σε περιβάλλον nested virtualization και όχι σε αυτόνομο φυσικό Proxmox host παραγωγικής εγκατάστασης. Η τελική διαμόρφωση περιλάμβανε δύο εικονικές μηχανές ανά pool. Το μέγεθος αυτό ήταν επαρκές για τον λειτουργικό έλεγχο της δημιουργίας, της ανάθεσης, της επαναφοράς, του συγχρονισμού και της αναπλήρωσης, αλλά δεν επιτρέπει την εξαγωγή συμπερασμάτων σχετικά με τη συμπεριφορά του συστήματος σε δεκάδες ή

εκατοντάδες ταυτόχρονες VMs.

Αντίστοιχα, ο έλεγχος παράλληλης ανάθεσης πραγματοποιήθηκε με δύο ταυτόχρονα αιτήματα. Το αποτέλεσμα επιβεβαίωσε ότι, στο συγκεκριμένο σενάριο, τα δύο αιτήματα εξυπηρετήθηκαν από διαφορετικές διαθέσιμες VMs μετά τη διόρθωση του μηχανισμού κλειδώματος. Δεν πραγματοποιήθηκε δοκιμή φόρτου με μεγάλο αριθμό ταυτόχρονων αιτημάτων και, επομένως, το αποτέλεσμα δεν αποτελεί μέτρηση της κλιμάκωσης ή της μέγιστης ρυθμαπόδοσης του broker.

Δεν πραγματοποιήθηκαν επίσης συστηματικές μετρήσεις χρόνου εκκίνησης, καθυστέρησης RDP/XRDP, κατανάλωσης CPU και μνήμης, δικτυακής απόδοσης ή χρόνου ολοκλήρωσης των lifecycle ενεργειών. Η επιτυχής απομακρυσμένη πρόσβαση χρησιμοποιήθηκε ως λειτουργικό κριτήριο και όχι ως μέτρηση της ποιότητας ή της απόδοσης της συνεδρίας.

Κατά τη δημιουργία και επαναφορά linked clones παρατηρήθηκαν προειδοποιήσεις του LVM-thin σχετικά με τη σχέση μεταξύ των λογικά δεσμευμένων μεγεθών και της διαθέσιμης φυσικής χωρητικότητας του thin pool. Οι προειδοποιήσεις δεν προκάλεσαν αποτυχία στις δοκιμές που πραγματοποιήθηκαν, ωστόσο καταδεικνύουν ότι η διαχείριση της πραγματικής χωρητικότητας αποθήκευσης αποτελεί παράμετρο που πρέπει να παρακολουθείται σε μεγαλύτερη εγκατάσταση. Δεν πραγματοποιήθηκε δοκιμή εξάντλησης του αποθηκευτικού χώρου.

Η χρήση παλαιότερων φυσικών υπολογιστών ως thin clients αποτελεί πιθανό σενάριο αξιοποίησης της αρχιτεκτονικής, αλλά δεν πραγματοποιήθηκε συστηματική πειραματική αξιολόγηση της απόδοσης ή της χρηστικότητας του τελικού thin-client περιβάλλοντος σε διαφορετικές κατηγορίες παλαιού υλικού. Για τον λόγο αυτό δεν εξάγεται σχετικό συμπέρασμα από τις παρούσες δοκιμές.

Τέλος, η περιοδική εκτέλεση της διαδικασίας επαναφοράς μέσω μηχανισμού προγραμματισμού, όπως cron ή systemd timer, καταγράφηκε ως δυνατότητα λειτουργικής αξιοποίησης αλλά δεν αποτέλεσε ανεξάρτητο σενάριο δοκιμής. Οι έλεγχοι του μηχανισμού επαναφοράς πραγματοποιήθηκαν μέσω άμεσης εκτέλεσης της αντίστοιχης διαδικασίας.

Με βάση τα παραπάνω, τα αποτελέσματα του κεφαλαίου τεκμηριώνουν τη λειτουργική ορθότητα των μηχανισμών που εξετάστηκαν στο συγκεκριμένο πειραματικό περιβάλλον. Δεν αποτελούν αξιολόγηση παραγωγικής κλιμάκωσης, μέγιστης χωρητικότητας ή συγκριτικής απόδοσης της υποδομής.

6.9 Σύνοψη Κεφαλαίου

Στο παρόν κεφάλαιο πραγματοποιήθηκε η λειτουργική επαλήθευση της τελικής υλοποίησης του συστήματος VDI. Οι δοκιμές κάλυψαν την πλήρη ροή πρόσβασης σε Windows και Linux περιβάλλοντα και τις βασικές λειτουργίες του κύκλου ζωής των διαχειριζόμενων εικονικών μηχανών.

Εκτός από τις αναμενόμενες επιτυχείς διαδρομές, εξετάστηκαν συνθήκες στις οποίες το σύστημα έπρεπε να αναβάλει ή να αποτρέψει μια ενέργεια. Η διαδικασία επαλήθευσης οδήγησε επίσης στον εντοπισμό προβλημάτων στον κοινό μηχανισμό κλειδώματος και στη διατήρηση της κατάστασης μετά από rollback, τα οποία διορθώθηκαν και επανελέγχθηκαν επιτυχώς πριν από την ολοκλήρωση της τελικής υλοποίησης.

Συνολικά, τα αποτελέσματα τεκμηριώνουν τη λειτουργική ορθότητα των μηχανισμών που εξετάστηκαν στο συγκεκριμένο πειραματικό περιβάλλον. Τα όρια της υποδομής και η απουσία

Πίνακας 6.2: Σύνοψη των βασικών λειτουργικών ελέγχων

Έλεγχος	Παρατηρούμενο αποτέλεσμα	Κατάσταση
Δημιουργία Linux και Windows pools	Δημιουργία των απαιτούμενων VMs και επίτευξη της επιθυμητής χωρητικότητας	Επιτυχής
Ανάθεση και απομακρυσμένη πρόσβαση	Επιτυχής σύνδεση σε Windows και Linux VM μέσω της πλήρους end-to-end ροής	Επιτυχής
Παράλληλη ανάθεση	Δύο ταυτόχρονα αιτήματα εξυπηρετήθηκαν από διαφορετικές Linux VMs μετά τη διόρθωση του κοινού μηχανισμού κλειδώματος	Επιτυχής
Προστασία ενεργής συνεδρίας	Ενεργές RDP/XRDP συνεδρίες απέτρεψαν την επαναφορά των αντίστοιχων VMs	Επιτυχής
Rollback και επαναχρησιμοποίηση	Επαναφορά στο <code>rdp-ready</code> και επιτυχής μεταγενέστερη ανάθεση	Επιτυχής
Επαναφορά ολόκληρου pool	Τα επιλέξιμα μέλη του Linux pool επανήλθαν στην καθαρή κατάσταση	Επιτυχής
Καταστροφή ολόκληρου Linux pool	Μετάβαση από 2/2 σε 0/2 χωρίς υπολειπόμενες καταχωρίσεις	Επιτυχής
Αναπλήρωση Linux pool	Αναγνώριση ελλείμματος και επαναφορά από 0/2 σε 2/2	Επιτυχής
Παρακολούθηση κατάστασης	Συνεπής απεικόνιση κατάστασης, snapshots, tags και locks	Επιτυχής

δοκιμών μεγάλης κλίμακας ή συστηματικών μετρήσεων απόδοσης δεν επιτρέπουν αντίστοιχα συμπεράσματα για παραγωγική κλιμάκωση, αλλά δεν επηρεάζουν τα λειτουργικά αποτελέσματα που επαληθεύτηκαν στο πλαίσιο της παρούσας εργασίας.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Εισαγωγή

Η παρούσα εργασία είχε ως αντικείμενο τη μελέτη, σχεδίαση και υλοποίηση ενός συστήματος αυτοματοποίησης της διαχείρισης υπολογιστικών πόρων με αξιοποίηση τεχνολογιών εικονικοποίησης και υποδομής VDI. Η ανάπτυξη πραγματοποιήθηκε με προσανατολισμό σε περιβάλλον εργαστηρίου υπολογιστών, όπου η επαναλαμβανόμενη προετοιμασία, διάθεση και επαναφορά σταθμών εργασίας δημιουργεί ανάγκη για κεντρική και αυτοματοποιημένη διαχείριση. Στα προηγούμενα κεφάλαια παρουσιάστηκαν διαδοχικά το θεωρητικό υπόβαθρο, οι τεχνολογίες που χρησιμοποιήθηκαν, η αρχιτεκτονική του συστήματος, η τελική υλοποίηση και η πειραματική επαλήθευσή της. Στο παρόν κεφάλαιο πραγματοποιείται η συνολική αποτίμηση του αποτελέσματος, προσδιορίζονται οι βασικοί περιορισμοί της υλοποίησης και παρουσιάζονται κατευθύνσεις μέσω των οποίων το σύστημα θα μπορούσε να επεκταθεί σε μελλοντική εργασία.

7.2 Συνολική Αποτίμηση της Εργασίας

Το τελικό σύστημα υλοποίησε μια ολοκληρωμένη διαδρομή διάθεσης εικονικών περιβαλλόντων εργασίας, από την επιλογή του επιθυμητού λειτουργικού συστήματος στον thin client έως την αυτόματη επιλογή και εκκίνηση κατάλληλης εικονικής μηχανής και την πραγματοποίηση απομακρυσμένης σύνδεσης. Η ίδια βασική διαδικασία εφαρμόστηκε σε δύο ανεξάρτητα pools, ένα για περιβάλλοντα Windows και ένα για Linux, επιτρέποντας την εξυπηρέτηση διαφορετικών κατηγοριών εργαστηριακής χρήσης μέσω κοινής υποδομής.

Παράλληλα, η διαχείριση των εικονικών μηχανών επεκτάθηκε πέρα από την απλή εκκίνηση και ανάθεσή τους. Υλοποιήθηκαν λειτουργίες δημιουργίας των απαιτούμενων μελών κάθε pool από προκαθορισμένα templates, επαναφοράς τους σε γνωστή καθαρή κατάσταση, ελεγχόμενης καταστροφής και μεταγενέστερης αναπλήρωσης της προκαθορισμένης χωρητικότητας. Με τον τρόπο αυτό, η διαχείριση μεταφέρθηκε από το επίπεδο της χειροκίνητης διαχείρισης μεμονωμένων VMs προς ένα μοντέλο στο οποίο τα μέλη αντιμετωπίζονται ως διαχειριζόμενοι και επαναχρησιμοποιήσιμοι πόροι ενός pool.

Ιδιαίτερη σημασία είχε η διατήρηση της ορθής κατάστασης του συστήματος κατά την εκτέλεση

των παραπάνω ενεργειών. Η τελική υλοποίηση περιλαμβάνει μηχανισμούς συγχρονισμού για την αποφυγή αντικρουόμενων ενεργειών στην ίδια VM, καθώς και ελέγχους που αποτρέπουν την επαναφορά εικονικών μηχανών κατά την περίοδο εκκίνησης ή ενώ εξυπηρετούν ενεργή απομακρυσμένη συνεδρία. Οι μηχανισμοί αυτοί επαληθεύτηκαν πειραματικά, συμπεριλαμβανομένης της εξυπηρέτησης δύο ταυτόχρονων αιτημάτων ανάθεσης από διαφορετικά διαθέσιμα μέλη του ίδιου pool.

Η πειραματική διαδικασία έδειξε επίσης ότι οι λειτουργίες του συστήματος μπορούν να συνδυαστούν σε έναν ολοκληρωμένο κύκλο ζωής. Εικονικές μηχανές διατέθηκαν σε clients, χρησιμοποιήθηκαν, επανήλθαν στο προκαθορισμένο snapshot και καταστάθηκαν ξανά διαθέσιμες. Σε επίπεδο ολόκληρου pool επαληθεύτηκε επιπλέον η πλήρης αφαίρεση των διαχειριζόμενων μελών και η μεταγενέστερη αναδημιουργία τους μέχρι την αποκατάσταση της προκαθορισμένης χωρητικότητας.

Η διαδικασία ανάπτυξης και επαλήθευσης ανέδειξε ταυτόχρονα τη σημασία της δοκιμής πραγματικών καταστάσεων λειτουργίας και όχι μόνο της επιτυχούς εκτέλεσης μεμονωμένων scripts. Οι δοκιμές αποκάλυψαν ζητήματα, όπως τα δικαιώματα πρόσβασης στον κοινό χώρο κλειδώματος και η διατήρηση της αναμενόμενης κατάστασης μετά από rollback, τα οποία διορθώθηκαν και επανελέγχθηκαν πριν από την ολοκλήρωση της τελικής διαμόρφωσης.

Συνολικά, η εργασία κατέληξε σε ένα λειτουργικό πρωτότυπο που συγκεντρώνει σε μία ενιαία ροή την προετοιμασία, διάθεση, προστασία, επαναφορά και αναπλήρωση εικονικών σταθμών εργασίας. Η υλοποίηση δεν επιχειρεί να αποτελέσει πλήρη πλατφόρμα VDI παραγωγικού επιπέδου, αλλά καταδεικνύει στην πράξη ότι βασικές διαδικασίες μαζικής διαχείρισης υπολογιστικών συστημάτων μπορούν να αυτοματοποιηθούν με τη συνδυασμένη αξιοποίηση του Proxmox VE, τυποποιημένων εικονικών μηχανών, μηχανισμών απομακρυσμένης πρόσβασης και ενός ελαφρού επιπέδου διαχείρισης και ανάθεσης.

7.3 Βασικά Αποτελέσματα και Συνεισφορά

Η βασική συνεισφορά της εργασίας βρίσκεται στη μετατροπή ενός συνόλου επιμέρους τεχνολογιών εικονικοποίησης, απομακρυσμένης πρόσβασης και αυτοματισμού σε ένα ενιαίο λειτουργικό σύστημα διαχείρισης εικονικών σταθμών εργασίας. Η αξιοποίηση του Proxmox VE ως υποδομής εικονικοποίησης συνδυάστηκε με μηχανισμούς αυτοματοποιημένης δημιουργίας, ανάθεσης, επαναφοράς και καταστροφής εικονικών μηχανών, καθώς και με έναν απλοποιημένο thin client μέσω του οποίου ο τελικός χρήστης μπορεί να αποκτήσει πρόσβαση στο επιθυμητό περιβάλλον εργασίας.

Ένα σημαντικό αποτέλεσμα της υλοποίησης ήταν η μετάβαση από τη διαχείριση συγκεκριμένων εικονικών μηχανών στη διαχείριση λογικών pools. Ο χρήστης δεν απαιτείται να γνωρίζει ποια συγκεκριμένη VM θα χρησιμοποιήσει, ενώ οι λειτουργίες διαχείρισης μπορούν να εφαρμοστούν σε επίπεδο pool και όχι αποκλειστικά σε μεμονωμένα VMIDs. Η προσέγγιση αυτή επιτρέπει τον διαχωρισμό μεταξύ του αιτήματος για έναν υπολογιστικό πόρο και της συγκεκριμένης εικονικής μηχανής που τελικά τον παρέχει.

Παράλληλα, η χρήση προκαθορισμένων templates και snapshots επέτρεψε τη διατήρηση μιας γνωστής κατάστασης αναφοράς για τα διαχειριζόμενα συστήματα. Η δυνατότητα επαναφοράς των VMs σε αυτή την κατάσταση και της μεταγενέστερης επαναχρησιμοποίησής τους αποτελεί ιδιαίτερα σημαντικό χαρακτηριστικό για εργαστηριακά περιβάλλοντα, στα οποία διαφο-

ρετικοί χρήστες χρησιμοποιούν διαδοχικά τους ίδιους υπολογιστικούς πόρους και είναι επιθυμητή η επαναφορά τους σε κοινή αρχική κατάσταση.

Επιπλέον, η εργασία δεν περιορίστηκε στην αυτοματοποίηση της κανονικής ροής λειτουργίας. Στην τελική υλοποίηση ενσωματώθηκαν μηχανισμοί προστασίας για καταστάσεις στις οποίες διαφορετικές λειτουργίες ενδέχεται να εκτελούνται ταυτόχρονα ή μια VM εξακολουθεί να χρησιμοποιείται. Η πειραματική επαλήθευση των μηχανισμών αυτών, καθώς και η διόρθωση προβλημάτων που εντοπίστηκαν κατά τις δοκιμές, συνέβαλαν ώστε το τελικό αποτέλεσμα να αποτελεί ολοκληρωμένο λειτουργικό πρωτότυπο και όχι απλώς ένα σύνολο ανεξάρτητων scripts. Τέλος, η υλοποίηση βασίστηκε σε διαθέσιμες τεχνολογίες και εργαλεία ανοικτού λογισμικού και διαμορφώθηκε με τρόπο που επιτρέπει την προσαρμογή των βασικών παραμέτρων του περιβάλλοντος, όπως τα templates, τα μεγέθη και τα εύρη VMIDs των pools. Η εργασία επομένως παρέχει ένα πρακτικό παράδειγμα του τρόπου με τον οποίο μπορεί να οργανωθεί ένα ελαφρύ σύστημα VDI για την αυτοματοποίηση επαναλαμβανόμενων εργασιών διαχείρισης σε ένα ελεγχόμενο εργαστηριακό περιβάλλον.

7.4 Περιορισμοί της Παρούσας Υλοποίησης

Παρά την επιτυχή λειτουργική ολοκλήρωση του πρωτοτύπου, η τελική υλοποίηση διατηρεί ορισμένους περιορισμούς που προκύπτουν από τον σκοπό και την έκταση της παρούσας εργασίας. Οι περιορισμοί αυτοί δεν αφορούν την πειραματική επαλήθευση που εξετάστηκε στο προηγούμενο κεφάλαιο, αλλά τις δυνατότητες που παρέχει το ίδιο το σύστημα στην παρούσα μορφή του.

Αρχικά, η υλοποίηση βασίζεται σε έναν κεντρικό Proxmox VE host, ο οποίος φιλοξενεί τόσο τις διαχειριζόμενες εικονικές μηχανές όσο και τις βασικές λειτουργίες του broker. Δεν έχει υλοποιηθεί κατανομή των VDI πόρων σε πολλαπλούς hosts ούτε μηχανισμός υψηλής διαθεσιμότητας. Κατά συνέπεια, η αρχιτεκτονική είναι κατάλληλη για το πειραματικό περιβάλλον της εργασίας, αλλά δεν αντιμετωπίζει περιπτώσεις αστοχίας του κεντρικού host ή ανάγκης κατανομής μεγάλου αριθμού VMs σε περισσότερους φυσικούς εξυπηρετητές.

Επιπλέον, το μέγεθος κάθε pool καθορίζεται μέσω της κεντρικής διαμόρφωσης και η διαδικασία δημιουργίας επιδιώκει την επίτευξη αυτής της προκαθορισμένης χωρητικότητας. Δεν έχει υλοποιηθεί δυναμική προσαρμογή του αριθμού των διαθέσιμων VMs με βάση τον αριθμό των ενεργών χρηστών, την τρέχουσα ζήτηση ή τη χρησιμοποίηση των πόρων του host. Η αναπλήρωση που υποστηρίζει το σύστημα αφορά επομένως την αποκατάσταση ενός δηλωμένου μεγέθους pool και όχι μηχανισμό αυτόματης κλιμάκωσης βάσει φορτίου.

Η διαχείριση των χρηστών παραμένει επίσης απλοποιημένη. Η πρόσβαση στα περιβάλλοντα εργασίας βασίζεται σε προκαθορισμένους τοπικούς λογαριασμούς των εικονικών μηχανών και δεν έχει ενσωματωθεί κεντρική υπηρεσία ταυτοποίησης χρηστών ή αντιστοίχιση μιας συνεδρίας VDI σε ατομικό πανεπιστημιακό λογαριασμό. Η επιλογή αυτή είναι συμβατή με τον στόχο ενός επαναχρησιμοποιήσιμου εργαστηριακού περιβάλλοντος, περιορίζει όμως δυνατότητες όπως η εξατομίκευση των συνεδριών, η εφαρμογή πολιτικών ανά χρήστη και η διατήρηση προσωπικών δεδομένων μεταξύ διαφορετικών VMs.

Αντίστοιχα, το σύστημα επικεντρώνεται στη διάθεση προσωρινών και επαναφερόμενων περιβαλλόντων εργασίας. Η επαναφορά στο προκαθορισμένο snapshot απομακρύνει τις μεταβολές που πραγματοποιήθηκαν κατά τη χρήση της VM και δεν έχει υλοποιηθεί ανεξάρτητος μηχανισμός.

νισμός διατήρησης προσωπικού προφίλ ή αρχείων του χρήστη. Σε ένα περιβάλλον στο οποίο απαιτείται μόνιμη αποθήκευση δεδομένων, αυτή θα πρέπει να παρέχεται από εξωτερική υπηρεσία ή από πρόσθετο μηχανισμό που δεν αποτελεί μέρος της παρούσας υλοποίησης.

Η παρακολούθηση και διαχείριση του συστήματος πραγματοποιείται κυρίως μέσω εργαλείων γραμμής εντολών και των υφιστάμενων δυνατοτήτων του Proxmox VE. Παρότι η εντολή `vdi-status` παρέχει συγκεντρωτική εικόνα της λειτουργικής κατάστασης των pools, δεν έχει αναπτυχθεί ξεχωριστό γραφικό ή διαδικτυακό περιβάλλον διαχείρισης για τις λειτουργίες του broker και του lifecycle των VMs.

Τέλος, η αξιοποίηση linked clones και thin-provisioned αποθηκευτικού χώρου μειώνει την ανάγκη για άμεση φυσική δέσμευση ολόκληρου του λογικού μεγέθους κάθε εικονικού δίσκου, χωρίς όμως να καταργεί τους περιορισμούς της πραγματικής αποθηκευτικής υποδομής. Η διατήρηση επαρκούς διαθέσιμης χωρητικότητας και η παρακολούθηση του thin pool παραμένουν ευθύνη του διαχειριστή. Η παρούσα υλοποίηση δεν περιλαμβάνει αυτοματοποιημένο μηχανισμό διαχείρισης ή επέκτασης της φυσικής αποθηκευτικής χωρητικότητας.

Οι παραπάνω περιορισμοί οριοθετούν το εύρος του πρωτοτύπου χωρίς να αναιρούν τον βασικό στόχο της εργασίας. Το σύστημα επικεντρώνεται στην αυτοματοποίηση του κύκλου ζωής και της διάθεσης επαναχρησιμοποιήσιμων εικονικών σταθμών εργασίας και όχι στην υλοποίηση όλων των υπηρεσιών που θα απαιτούσε μια ολοκληρωμένη πλατφόρμα VDI παραγωγικού επιπέδου.

7.5 Μελλοντικές Επεκτάσεις

Η υλοποίηση που αναπτύχθηκε μπορεί να αποτελέσει βάση για περαιτέρω επέκταση προς ένα πληρέστερο σύστημα διαχείρισης εικονικών εργαστηριακών υποδομών. Οι πιθανές επεκτάσεις αφορούν τόσο την κλιμάκωση της υποδομής όσο και την προσθήκη λειτουργιών διαχείρισης χρηστών, παρακολούθησης και αυτοματοποίησης.

Μια βασική κατεύθυνση αποτελεί η επέκταση από έναν μεμονωμένο Proxmox VE host σε υποδομή πολλαπλών hosts. Σε μια τέτοια διαμόρφωση, ο broker θα μπορούσε να λαμβάνει υπόψη τη διαθεσιμότητα και τη χρησιμοποίηση των πόρων κάθε κόμβου πριν από τη δημιουργία ή την ανάθεση μιας εικονικής μηχανής. Παράλληλα, η αξιοποίηση δυνατοτήτων clustering και υψηλής διαθεσιμότητας θα μπορούσε να μειώσει την εξάρτηση της υπηρεσίας από έναν μοναδικό φυσικό εξυπηρετητή[30].

Μια δεύτερη επέκταση αφορά τη δυναμική προσαρμογή της χωρητικότητας των pools. Στην παρούσα υλοποίηση κάθε pool διαθέτει προκαθορισμένο επιθυμητό μέγεθος. Μελλοντικά, ο αριθμός των διαθέσιμων VMs θα μπορούσε να μεταβάλλεται σύμφωνα με την πραγματική ζήτηση, τον αριθμό των αιτημάτων ή τη διαθέσιμη υπολογιστική χωρητικότητα. Ένας τέτοιος μηχανισμός θα μπορούσε να αυξάνει προσωρινά το μέγεθος ενός pool όταν αυξάνεται η ζήτηση και να απομακρύνει πλεονάζοντα ανενεργά μέλη όταν αυτή μειώνεται, διατηρώντας παράλληλα τους απαραίτητους ελέγχους προστασίας του lifecycle.

Ιδιαίτερο ενδιαφέρον παρουσιάζει επίσης η ενσωμάτωση κεντρικού μηχανισμού ταυτοποίησης και διαχείρισης χρηστών. Η σύνδεση με μια υπηρεσία καταλόγου ή με την υφιστάμενη υποδομή ταυτοποίησης ενός οργανισμού θα επέτρεπε την αντιστοίχιση των αιτημάτων σε συγκεκριμένους χρήστες και την εφαρμογή διαφορετικών πολιτικών πρόσβασης. Σε συνδυασμό με εξωτερικό αποθηκευτικό χώρο, θα μπορούσε επιπλέον να παρέχεται διατήρηση προσωπι-

κών αρχείων ή προφίλ ανεξάρτητα από την προσωρινή VM που ανατίθεται σε κάθε συνεδρία. Η διαχείριση του συστήματος θα μπορούσε επίσης να επεκταθεί μέσω ενός κεντρικού γραφικού ή διαδικτυακού περιβάλλοντος. Ένα τέτοιο περιβάλλον θα μπορούσε να παρουσιάζει την κατάσταση των pools, τις ενεργές συνεδρίες, τη διαθεσιμότητα των VMs και τις βασικές πληροφορίες της υποδομής, ενώ παράλληλα θα μπορούσε να επιτρέπει την ελεγχόμενη εκτέλεση ενεργειών δημιουργίας, επαναφοράς, καταστροφής και αναπλήρωσης χωρίς άμεση χρήση της γραμμής εντολών.

Περαιτέρω αυτοματοποίηση θα μπορούσε να εφαρμοστεί και στη χρονική οργάνωση των lifecycle λειτουργιών. Η περιοδική επαναφορά των κατάλληλων VMs μπορεί να ενσωματωθεί σε συστηματικά προγραμματισμένη διαδικασία, ενώ διαφορετικές πολιτικές θα μπορούσαν να εφαρμόζονται ανά pool ή ανά χρονική περίοδο. Σε ένα εκπαιδευτικό περιβάλλον, για παράδειγμα, η προετοιμασία και επαναφορά συγκεκριμένων pools θα μπορούσε να συνδέεται με το πρόγραμμα μαθημάτων ή εξετάσεων.

Επιπλέον, μια μελλοντική έκδοση θα μπορούσε να ενσωματώσει πληρέστερη παρακολούθηση της χρήσης των διαθέσιμων πόρων. Μετρήσεις για CPU, μνήμη, αποθηκευτικό χώρο, αριθμό ενεργών VMs και κατάσταση του thin-provisioned storage θα μπορούσαν να χρησιμοποιηθούν τόσο για την ενημέρωση του διαχειριστή όσο και ως είσοδοι για αυτοματοποιημένες αποφάσεις διαχείρισης. Ιδιαίτερα η παρακολούθηση της πραγματικής κατανάλωσης του αποθηκευτικού χώρου θα μπορούσε να συνοδεύεται από προειδοποιήσεις πριν η διαθέσιμη φυσική χωρητικότητα προσεγγίσει κρίσιμα επίπεδα.

Τέλος, σημαντική συνέχεια της παρούσας εργασίας θα αποτελούσε η εφαρμογή και αξιολόγηση του συστήματος σε μεγαλύτερη φυσική εργαστηριακή εγκατάσταση. Η χρήση πραγματικών σταθμών εργασίας ως thin clients, μεγαλύτερων pools και αυξημένου αριθμού ταυτόχρονων χρηστών θα επέτρεπε τη συλλογή ποσοτικών δεδομένων για χρόνους ανάθεσης και εκκίνησης, κατανάλωση πόρων, δικτυακή συμπεριφορά και ποιότητα απομακρυσμένης πρόσβασης. Με αυτόν τον τρόπο θα μπορούσε να αξιολογηθεί όχι μόνο η λειτουργική ορθότητα, αλλά και η πρακτική κλιμάκωση της προσέγγισης υπό συνθήκες πλησιέστερες σε πραγματική παραγωγική χρήση.

Οι παραπάνω επεκτάσεις δεν είναι απαραίτητες για τη λειτουργία του πρωτοτύπου που αναπτύχθηκε, αλλά αποτελούν φυσικές κατευθύνσεις εξέλιξής του. Η αρθρωτή οργάνωση της υλοποίησης και ο διαχωρισμός μεταξύ thin client, broker, pools και lifecycle λειτουργιών παρέχουν σημεία στα οποία μπορούν να προστεθούν νέες δυνατότητες χωρίς να απαιτείται αναγκαστικά πλήρης επανασχεδίαση του συστήματος.

7.6 Τελικό Συμπέρασμα

Η παρούσα εργασία διερεύνησε την εφαρμογή τεχνολογιών εικονικοποίησης και VDI ως βάση για την αυτοματοποίηση της διαχείρισης υπολογιστικών συστημάτων σε εργαστηριακό περιβάλλον. Από την αρχική μελέτη του προβλήματος προέκυψε σταδιακά ένα λειτουργικό πρωτότυπο στο οποίο η προετοιμασία, η διάθεση και η επαναφορά εικονικών σταθμών εργασίας μπορούν να πραγματοποιούνται κεντρικά και με περιορισμένη ανάγκη χειροκίνητης παρέμβασης.

Η τελική υλοποίηση υποστήριξε δύο ανεξάρτητα pools εικονικών μηχανών για περιβάλλοντα Windows και Linux και ολοκλήρωσε σε κοινή ροή τη δημιουργία των απαιτούμενων πόρων,

την επιλογή και ανάθεσή τους μέσω του broker, την απομακρυσμένη πρόσβαση από thin client και τις βασικές λειτουργίες διαχείρισης του κύκλου ζωής τους. Παράλληλα, ενσωματώθηκαν μηχανισμοί προστασίας για την αποφυγή αντικρουόμενων ενεργειών και την προστασία ενεργών συνεδριών. Η πειραματική διαδικασία επιβεβαίωσε τη λειτουργία των βασικών αυτών μηχανισμών στο περιβάλλον στο οποίο αναπτύχθηκε το σύστημα.

Το αποτέλεσμα δεν αποτελεί ολοκληρωμένη πλατφόρμα VDI παραγωγικού επιπέδου και διατηρεί περιορισμούς ως προς την κλιμάκωση, τη διαχείριση ταυτότητας χρηστών, την υψηλή διαθεσιμότητα και την κεντρική παρακολούθηση. Ωστόσο, αποδεικνύει σε επίπεδο λειτουργικού πρωτοτύπου ότι ένα σημαντικό μέρος των επαναλαμβανόμενων ενεργειών που σχετίζονται με τη διάθεση και διαχείριση εργαστηριακών υπολογιστικών περιβαλλόντων μπορεί να μεταφερθεί από τη χειροκίνητη διαχείριση σε αυτοματοποιημένες και επαναλήψιμες διαδικασίες. Υπό αυτή την έννοια, η εργασία συνδέεται άμεσα με τον κεντρικό στόχο της αυτοματοποίησης της μαζικής διαχείρισης υπολογιστικών συστημάτων. Η προσέγγιση που υλοποιήθηκε αντιμετωπίζει τις εικονικές μηχανές όχι ως μεμονωμένους σταθμούς που απαιτούν ανεξάρτητη διαχείριση, αλλά ως μέλη διαχειριζόμενων pools που μπορούν να δημιουργούνται, να διατίθενται, να επαναφέρονται και να αναπληρώνονται με κοινές διαδικασίες. Το λειτουργικό πρωτότυπο που προέκυψε αποτελεί επομένως μια πρακτική απόδειξη της δυνατότητας εφαρμογής αυτής της προσέγγισης και μια βάση πάνω στην οποία μπορούν να αναπτυχθούν πληρέστερες και μεγαλύτερης κλίμακας λύσεις στο μέλλον.

Βιβλιογραφία

- [1] IBM. "What Is Virtualization?", επίσκεψη 12 Αύγ. 2026. διεύθυν.: <https://www.ibm.com/think/topics/virtualization>
- [2] Red Hat. "General Introduction to Virtualization", Red Hat Enterprise Linux Documentation, επίσκεψη 12 Αύγ. 2026. διεύθυν.: https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/virtualization_getting_started_guide/chap-virtualization_getting_started-what_is_it
- [3] Red Hat. "Guest Virtual Machine Device Configuration", Red Hat Enterprise Linux 7 Virtualization Deployment και Administration Guide, επίσκεψη 12 Αύγ. 2026. διεύθυν.: https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/chap-guest_virtual_machine_device_configuration
- [4] Red Hat. "What is a Hypervisor?", επίσκεψη 12 Αύγ. 2026. διεύθυν.: <https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor>
- [5] IBM. "What Are Hypervisors?", επίσκεψη 12 Αύγ. 2026. διεύθυν.: <https://www.ibm.com/think/topics/hypervisors>
- [6] Intel Corporation. "Intel Virtualization Technology (Intel VT) for Intel 64 and Intel Architecture (Intel VT-x)", Intel, επίσκεψη 12 Αύγ. 2026. διεύθυν.: <https://edc.intel.com/content/www/us/en/design/products-and-solutions/processors-and-chipsets/core-ultra-200h-and-200u-series-processors-datasheet-volume-1-of-2/intel-virtualization-technology-intel-vt-for-intel-64-and-intel-architecture-int/>
- [7] Advanced Micro Devices, Inc., *AMD64 Architecture Programmer's Manual, Volume 2: System Programming*, Revision 3.44, 24593, See Chapter 15, Secure Virtual Machine (SVM), Advanced Micro Devices, Inc., Μαρ. 2026. επίσκεψη 4 Σεπτ. 2026. διεύθυν.: https://docs.amd.com/v/u/en-US/24593_3.45_APM_Vol2_PUB
- [8] IBM. "What Is Desktop Virtualization?", επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.ibm.com/think/topics/desktop-virtualization>
- [9] IBM. "What Is Virtual Desktop Infrastructure (VDI)?", επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.ibm.com/think/topics/virtual-desktop-infrastructure>
- [10] A. A. Z. A. Ibrahim, D. Kliazovich, P. Bouvry και A. Oleksiak, "Virtual Desktop Infrastructures: Architecture, Survey and Green Aspects Proof of Concept", στο *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, 2016, σσ. 269–276. doi:10.1109/IGCC.2016.7892624

- [11] Microsoft. "Device Identity and Desktop Virtualization", Microsoft Learn, επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://learn.microsoft.com/en-us/entra/identity/devices/howto-device-identity-virtual-desktop-infrastructure>
- [12] T. J. M. Sanguino, I. F. de Viana, D. A. L. García και E. C. Ancos, "OpenGnSys: A Novel System toward Centralized Deployment and Management of Computer Laboratories", *Computers & Education*, τόμ. 75, σσ. 30–43, 2014. doi: 10.1016/j.compedu.2014.01.011
- [13] H. N. Mok, Y. L. Lee και W. K. Tan, "Setting Up a Low-cost Lab Management System for a Multi-purpose Computing Laboratory Using Virtualisation Technology", *Australasian Journal of Educational Technology*, τόμ. 28, αρθμ. 2, 2012. doi: 10.14742/ajet.873
- [14] A. Johnson, K. Dempsey, R. Ross, S. Gupta και D. Bailey, "Guide for Security-Focused Configuration Management of Information Systems", National Institute of Standards και Technology, NIST Special Publication 800-128, 2019. doi: 10.6028/NIST.SP.800-128
- [15] Red Hat. "What Is IT Automation?", Red Hat, επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.redhat.com/en/topics/automation/what-is-it-automation>
- [16] Red Hat. "What Is Configuration Management?", Red Hat, επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.redhat.com/en/topics/automation/what-is-configuration-management>
- [17] S. J. Yang, J. Nieh, M. Selsky και N. Tiwari, "The Performance of Remote Display Mechanisms for Thin-Client Computing", στο *2002 USENIX Annual Technical Conference*, Monterey, CA: USENIX Association, 2002.
- [18] Microsoft. "[MS-RDPBCGR] Remote Desktop Protocol: Basic Connectivity and Graphics Remoting", Microsoft Open Specifications, επίσκεψη 13 Αύγ. 2026. διεύθυν.: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/5073f4ed-1e93-45e1-b039-6e30c385867c
- [19] Microsoft. "[MS-RDPEGFX] Remote Desktop Protocol: Graphics Pipeline Extension", Microsoft Open Specifications, επίσκεψη 13 Αύγ. 2026. διεύθυν.: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpegfx/0926f8c8-33e1-44ef-baee-3446f2192b0f
- [20] Microsoft. "[MS-RDPBCGR] Applicability Statement", Microsoft Open Specifications, επίσκεψη 13 Αύγ. 2026. διεύθυν.: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/5059e70b-e859-4eb8-b0f5-7f320f1f8ae9
- [21] Microsoft. "[MS-RDPBCGR] Enhanced RDP Security", Microsoft Open Specifications, επίσκεψη 13 Αύγ. 2026. διεύθυν.: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/592a0337-dc91-4de3-a901-e1829665291d
- [22] T. Richardson και J. Levine, "The Remote Framebuffer Protocol", Internet Engineering Task Force, RFC 6143, Μαρ. 2011. doi: 10.17487/RFC6143 επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.rfc-editor.org/rfc/rfc6143>

- [23] SPICE Project. "SPICE Remote Computing Protocol", SPICE Project, επίσκεψη 13 Αύγ. 2026. διεύθυν.: https://www.spice-space.org/static/docs/spice_protocol.pdf
- [24] IBM. "What Is a Thin Client?", IBM, επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.ibm.com/think/topics/thin-client>
- [25] J. Martínez-Mateo, S. Munoz-Hernandez και D. Pérez-Rey, "A DISCUSSION OF THIN CLIENT TECHNOLOGY FOR COMPUTER LABS", στο *Proceedings of the Multi-Conference on Innovative Developments in ICT - Volume 1: ICTEL, (INNOV 2010)*, INSTICC, SciTePress, 2010, σσ. 119–124, isbn: 978-989-8425-15-7. doi: 10.5220/0002962901190124
- [26] Proxmox Server Solutions GmbH. "Proxmox Virtual Environment - Features", επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://www.proxmox.com/en/products/proxmox-virtual-environment/features>
- [27] Proxmox Server Solutions GmbH, *Proxmox VE Administration Guide*, Proxmox Server Solutions GmbH. επίσκεψη 13 Αύγ. 2026. διεύθυν.: <https://pve.proxmox.com/pve-docs/pve-admin-guide.html>
- [28] Proxmox Server Solutions GmbH. "Proxmox VE Network Configuration", επίσκεψη 14 Αύγ. 2026. διεύθυν.: https://pve.proxmox.com/pve-docs/chapter-sysadmin.html#sysadmin_network_configuration
- [29] Proxmox Server Solutions GmbH. "Proxmox VE Storage", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://pve.proxmox.com/pve-docs/chapter-pvesm.html>
- [30] Proxmox Server Solutions GmbH. "Proxmox VE Cluster Manager", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://pve.proxmox.com/pve-docs/chapter-pvecm.html>
- [31] The Linux Kernel Documentation. "The Definitive KVM (Kernel-based Virtual Machine) API Documentation", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://docs.kernel.org/virt/kvm/api.html>
- [32] Proxmox Server Solutions GmbH. "QEMU/KVM Virtual Machines", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://pve.proxmox.com/pve-docs/chapter-qm.html>
- [33] QEMU Project. "QEMU Documentation", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://www.qemu.org/docs/master/>
- [34] Proxmox Server Solutions GmbH. "qm -- QEMU/KVM Virtual Machine Manager", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://pve.proxmox.com/pve-docs/qm.1.html>
- [35] Proxmox Server Solutions GmbH. "Proxmox VE API", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://pve.proxmox.com/pve-docs/api-viewer/>
- [36] QEMU Project. "QEMU Guest Agent", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://www.qemu.org/docs/master/interop/qemu-ga.html>
- [37] virtio-win Project. "Windows VirtIO Drivers", επίσκεψη 14 Αύγ. 2026. διεύθυν.: <https://github.com/virtio-win/virtio-win-pkg-scripts>

- [38] Microsoft. "Sysprep (System Preparation) Overview", επίσκεψη 14 Αύγ. 2026. διεύθν.: <https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/sysprep--system-preparation--overview>
- [39] Microsoft. "Answer Files (unattend.xml)", επίσκεψη 14 Αύγ. 2026. διεύθν.: <https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/update-windows-settings-and-scripts-create-your-own-answer-file-sxs>
- [40] FreeRDP Project. "FreeRDP", επίσκεψη 15 Αύγ. 2026. διεύθν.: <https://github.com/FreeRDP/FreeRDP>
- [41] The xrdp Project. "xrdp: An Open Source RDP Server", επίσκεψη 14 Αύγ. 2026. διεύθν.: <https://github.com/neutrinolabs/xrdp>
- [42] The xrdp Project. "xorgxrdp", επίσκεψη 14 Αύγ. 2026. διεύθν.: <https://github.com/neutrinolabs/xorgxrdp>
- [43] Debian Project. "About Debian", επίσκεψη 15 Αύγ. 2026. διεύθν.: <https://www.debian.org/intro/about>
- [44] X.Org Foundation. "X.Org Documentation", επίσκεψη 14 Αύγ. 2026. διεύθν.: <https://www.x.org/wiki/Documentation/>
- [45] X.Org Foundation, *startx -- Initialize an X Session*. επίσκεψη 4 Σεπτ. 2026. διεύθν.: <https://www.x.org/docs/man/man.pdf>
- [46] OpenBSD Project. "ssh -- OpenSSH Remote Login Client", επίσκεψη 15 Αύγ. 2026. διεύθν.: <https://man.openbsd.org/ssh>
- [47] OpenBSD Project. "OpenSSH", επίσκεψη 15 Αύγ. 2026. διεύθν.: <https://www.openssh.com/>
- [48] OpenBSD Project. "sshd -- OpenSSH Daemon: Authorized Keys", επίσκεψη 15 Αύγ. 2026. διεύθν.: https://man.openbsd.org/sshd#AUTHORIZED_KEYS_FILE_FORMAT
- [49] National Institute of Standards and Technology. "Least Privilege", επίσκεψη 15 Αύγ. 2026. διεύθν.: https://csrc.nist.gov/glossary/term/least_privilege
- [50] Free Software Foundation. "Bash Reference Manual", επίσκεψη 15 Αύγ. 2026. διεύθν.: <https://www.gnu.org/software/bash/manual/bash.html>
- [51] util-linux Project. "flock(1) -- Manage Locks from Shell Scripts", επίσκεψη 16 Αύγ. 2026. διεύθν.: <https://man7.org/linux/man-pages/man1/flock.1.html>