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

Εικονικά περιβάλλοντα και τεχνικές ενοποίησής τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Λέκκας

Επιβλέπων : Βασίλειος Μάγκλαρης
Καθηγητής ΕΜΠ

Αθήνα, Νοέμβριος 2011

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

Εικονικά περιβάλλοντα και τεχνικές ενοποίησής τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Λέκκας

Επιβλέπων : Βασίλειος Μάγκλαρης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Νοεμβρίου 2011.

.....	.....	.....
Βασίλειος Μάγκλαρης	Συμεών Παπαβασιλείου	Δημήτριος Καλογεράς
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Ερευνητής ΕΠΙΣΕΥ

Αθήνα, Νοέμβριος 2011

.....

Κωνσταντίνος Λ. Λέκκας

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

Περίληψη

Το διαδίκτυο με τη σημερινή μορφή του θέτει κάποιους περιορισμούς στην ανάπτυξη καινοτόμων αρχιτεκτονικών και υπηρεσιών. Προκειμένου να αντιμετωπιστούν οι συγκεκριμένοι περιορισμοί αρχιτεκτονικών αποφασίστηκε η εκτέλεση των νέων τεχνολογιών σε πειραματικό επίπεδο σε πειραματικές πλατφόρμες – εικονικά δίκτυα παράλληλα με τις ήδη υπάρχουσες αρχιτεκτονικές με χρήση τεχνικών εικονικοποίησης, μιας μεθόδου που υφίσταται αρκετά χρόνια στον χώρο των τεχνολογιών πληροφορικής και επικοινωνιών προσφέροντας πλήρη απομόνωση μεταξύ των εκτελούμενων στις υπάρχουσες υποδομές υπηρεσιών και αρχιτεκτονικών. Η εφαρμογή της εικονικοποίησης εισάγει τα «νέφη υπολογιστών» που στηρίζουν την λειτουργία τους σε εικονικά δίκτυα που βασίζονται στο λογισμικό. Το ερευνητικό ενδιαφέρον εστιάζεται στην ενοποίηση διαφορετικών πλατφορμών και υπηρεσιών με στόχο τον αποτελεσματικότερο έλεγχο τους. Στη παρούσα διπλωματική εργασία αφού αναλυθούν διεξοδικά οι κατηγορίες και τεχνολογίες εικονικοποίησης περιγράφονται τεχνολογίες ενοποίησης πειραματικών πλατφορμών και πως αυτές έχουν εφαρμοστεί σε ερευνητικό επίπεδο όπως στο NOVI Project.

Λέξεις Κλειδιά: εικονικοποίηση, εικονικά δίκτυα, πειραματικές πλατφόρμες, ενοποίηση

Abstract

Internet in its current form poses some restrictions on the development of innovative architectures and services. In order for these restrictions to be avoided, it was decided that new technologies - methods will be executed simultaneously with the use of old ones in an experimental level in testbeds - virtual networks. These methods use the technique of virtualization, which is commonly used the last years in the era of information and communications technology and offer complete isolation between the innovative architectures and services and the existing ones. The use of virtualization introduces cloud computing whose functionality is relied on virtual networks that are software based. The researchers focus their interest on the federation of different platforms and services to control them more efficiently. In this diploma thesis, after describing thoroughly the virtualization methods, we describe federation technologies that federate testbeds and also we examine how these technologies are applied in a research level such as NOVI Project.

Keywords: virtualization, virtual networks, testbeds, federation

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Βασίλειο Μάγκλαρη για την ανάθεση της παρούσας διπλωματικής εργασίας. Ακόμη, ευχαριστώ όλα τα μέλη του Εργαστηρίου Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων για τη βοήθειά τους, και ειδικότερα την υπεύθυνη της διπλωματικής εργασίας κα Μαίρη Γραμματικού. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την υποστήριξη που μου προσέφεραν σε ολόκληρη τη διάρκεια των σπουδών μου.

Κωνσταντίνος Λέκκας

Περιεχόμενα

1. Εισαγωγή

1.1 Future Internet και virtualization

Παρά την εντατική έρευνα των τελευταίων ετών το internet συνεχίζει να βασίζεται στην αρχή της βέλτιστης προσπάθειας όπως εκείνη προκύπτει από το IP πρωτόκολλο. Ταυτόχρονα, κρίνεται επιτακτική η ανάγκη για έρευνα και εφαρμογή νέων τεχνολογιών. Από την άλλη πλευρά οι σύγχρονοι internet service providers είναι επιφυλακτικοί στην υιοθέτηση νέων τεχνολογιών για το δίκτυο τους υπό το φόβο της χειροτέρευσης των υπηρεσιών που ήδη προσφέρουν. Σε μια προσπάθεια να διασφαλιστεί η εύρωστη λειτουργία των υφιστάμενων δικτύων αλλά και να συνεχιστεί η ανάπτυξη καινοτόμων υπηρεσιών η αρχιτεκτονική του μελλοντικού internet θα είναι πολυμορφική με την έννοια ότι θα μπορούν να συνυπάρχουν σε αυτό η παρούσα δικτυακή αρχιτεκτονική μαζί με νέα πρωτόκολλα υπηρεσίες και αρχιτεκτονικές. Το Future Internet, όπως ακριβώς ονομάζεται το μελλοντικό διαδίκτυο, που θα βασίζει τη λειτουργία του στη συνύπαρξη πολλών αρχιτεκτονικών θα πρέπει, κατά συνέπεια, να δομηθεί με εφαρμογή τεχνολογιών εικονικοποίησης. Ως εικονικοποίηση (virtualization) νοείται η δυνατότητα του διαχωρισμού και καταμερισμού των πόρων ενός φυσικού κόμβου σε ένα πλήθος από εικονικούς κόμβους (virtual nodes). Ο πολυμορφικές αρχιτεκτονικές σε συνδυασμό με εικονικοποίηση δικτυακών πόρων και συστημάτων μεταβάλλει ολόκληρη τη φιλοσοφία πάνω στην οποία δομείται το διαδίκτυο. Με αυτό τον τρόπο η εικονικοποίηση καθίσταται καθοριστικό στοιχείο της έρευνας για το Future Internet επιτρέποντας την σχεδίαση και υιοθέτηση νέων τεχνολογιών παράλληλα με διατήρηση των ήδη υπαρχόντων.

1.2 Αντικείμενο διπλωματικής

Στην παρούσα διπλωματική εργασία θα γίνει μια ανασκόπηση των τεχνικών που χρησιμοποιούνται κατά την εικονικοποίηση συστήματων και δικτύων. Στην εικονικοποίηση θα βασιστεί η δομή του μελλοντικού διαδικτύου κατά συνέπεια έχει ιδιαίτερη σημασία η αποσαφήνιση των μεθοδολογιών και τεχνικών που θα μελετηθούν. Θα περιγραφούν βασικές πειραματικές πλατφόρμες καθώς και τεχνικές ενοποίησης τους.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν οι διάφορες κατηγορίες εικονικοποίησης
2. Παρουσιάστηκαν τα πλεονεκτήματα και τα μειονεκτήματα κάθε μεθόδου
3. Έγινε αναφορά στην τεχνολογία που βασίζονται οι διάφορες πειραματικές πλατφόρμες που έχουν αναπτυχθεί.
4. Αναφέρθηκαν μέθοδοι ενοποίησης πειραματικών πλατφόρμων

Οργάνωση κειμένου

Αρχικά, στο δεύτερο κεφάλαιο γίνεται αναφορά στην έννοια της εικονικοποίησης, μέσω μιας ιστορικής αναδρομή με παρουσίαση τεχνολογιών που εφαρμόστηκαν. Επιπλέον, γίνεται διάκριση μεταξύ των διάφορων κατηγοριών εικονικοποίησης υλικού (εικονικοποίηση συστήματος, δικτυακή εικονικοποίηση, εικονικοποίηση σε επίπεδο λειτουργικού συστήματος κλπ) αλλά και στις κατηγορίες εικονικοποίησης συστημάτων. Στο τρίτο κεφάλαιο της εργασίας γίνεται παρουσίαση των δομικών λίθων της εικονικοποίησης όπως των VMware εικονικών μηχανών για την πλήρη εικονικοποίηση και του Xen για την μερική εικονικοποίηση. Στο τέταρτο κεφάλαιο της εργασίας παρουσιάζονται ορισμένες πειραματικές υποδομές και τέλος στο πέμπτο κεφάλαιο γίνεται αναφορά στην SFA αρχιτεκτονική και στο Teagle καθώς επίσης και στο NOVI Project που στοχεύει στην ενοποίηση πειραματικών υποδομών.

2. Υπόβαθρο

2.1 Εικονικοποίηση

2.1.1 Εισαγωγή

Η εικονικοποίηση (virtualization) συνίσταται στον διαμοιρασμό των πόρων ενός υπάρχοντος φυσικού συστήματος σε πολλαπλές virtual machines. Συναντάται από τις απαρχές των υπολογιστών και βρίσκει εφαρμογές στους υπολογιστές, στα δίκτυα, στα μέσα αποθήκευσης προσωρινού (RAM) και μόνιμου χαρακτήρα (σκληροί δίσκοι, RAID) και στις εφαρμογές. Το network virtualization αποτελείται από δύο βασικά στοιχεία: link και node virtualization. Το link virtualization επιτρέπει τη μεταφορά πολλών ξεχωριστών εικονικών ζεύξεων πάνω από μία κοινή φυσική ζεύξη. Η εικονική ζεύξη χαρακτηρίζεται ρητά από μια ετικέτα ή μπορεί να αφορά, σε μια γενικότερη προσέγγιση, μια χρονοθυρίδα ή ένα μήκος κύματος. Στο διαδίκτυο, πρωτόκολλα όπως ATM και MPLS χρησιμοποιούνται για αυτό το σκοπό. Το node virtualization βασίζεται στο διαχωρισμό και την κατάτμηση των πόρων υλικού. Οι φυσικοί πόροι ενός κόμβου (CPU, μνήμη, χωρητικότητα, εύρος ζώνης) χωρίζονται σε slices και κάθε slice εκχωρείται στον εικονικό κόμβο σύμφωνα με ένα σύνολο απαιτήσεων. Ο συνδυασμός των δύο αυτών στοιχείων δημιουργεί τα εικονικά δίκτυα, τα οποία από λειτουργική άποψη είναι ίδια με τα φυσικά [10]. Η χρήση τεχνικών virtualization αυξάνεται σταθερά τα τελευταία χρόνια και καλύπτουν ένα μεγάλο εύρος του IT καλύπτοντας και το πεδίο των μέσων αποθήκευσης (storage virtualization).

2.1.2 Ιστορική Αναδρομή

Η χρήση των εικονικών μηχανημάτων (virtual machines) που συνιστούν τον δομικό λίθο ενός virtual περιβάλλοντος έγινε για πρώτη φόρα από την IBM το 1960 σε μια προσπάθεια μείωσης του κόστους σε πολυδάπανο υπολογιστικό εξοπλισμό. Στόχος ήταν η ταυτόχρονη εκτέλεση εφαρμογών και διαδικασιών. Κατά την διάρκεια των δεκαετιών του ’80 και του ’90 κυριαρχούσαν τα κατανεμημένα συστήματα, οι client-server εφαρμογές και οι x86server. Μόλις πρόσφατα με την αλματώδη πρόοδο της τεχνολογίας οι πολυπύρηνοι επεξεργαστές έκαναν έντονη την εμφάνιση τους στον χώρο του IT.

Στο παρελθόν έχουν διατυπωθεί και αναπτυχθεί ιδέες και τεχνολογίες που παραπέμπουν ή συνδέονται άμεσα με το virtualization: τα εικονικά δίκτυα περιοχής (Virtual Local Area Networks – VLANs) , τα εικονικά ιδιωτικά δίκτυα (Virtual Private Networks – VPNs) , τα προγραμματίσιμα δίκτυα (programmable networks) και τα overlay networks.

Link Virtualization:

Η εικονικοποίηση ζεύξεων (link virtualization) συνίστανται στον διαμοιρασμό του διαθέσιμου εύρους ζώνης σε λογικές συνδέσεις. Η εικονικοποίηση αυτή επιτυγχάνεται είτε με την ένταξη των συστημάτων που θέλουμε να διασυνδέσουμε στο ίδιο VLan είτε με τη δημιουργία VPNs.

VLans: Ένα VLan [1] είναι μια ομάδα από hosts που ανήκουν στην ίδια ομάδα εκπομπής ανε-ξάρτητα από το φυσικό τρόπο σύνδεσης τους. Όλα τα πλαίσια σε ένα VLan χαρακτηρίζονται από το ίδιο vlan id. Έτσι για παράδειγμα ενώ δύο τερματικά που είναι συνδεδεμένα σε δύο πόρτες ενός μεταγωγέα μπορεί φαινομενικά να ανήκουν στην ίδια ομάδα εκπομπής δηλαδή στο ίδιο LAN υπάρχει όμως η δυνατότητα να εντάξουμε το ένα σε μια άλλη ομάδα εκπομπής δημιουργώντας ουσιαστικά ένα δεύτερο LAN. Τα δύο LAN που προκύπτουν τελικά τα ονομάζουμε VLANs. Ο μεταγωγέας σε κάθε περίπτωση χρησιμοποιεί την διεύθυνση MAC του προορισμού και το vlan id προκειμένου να προωθήσει τα πλαίσια. Στην ουσία εισάγεται ένα επιπλέον επίπεδο πολυπλεξίας.
Οι λόγοι που καθιστούν αναγκαίο πολλές φορές τον σχεδιασμό VLANs είναι οι εξής [2]:

- Η δημιουργία μικρότερων LAN συνεπάγεται μείωση του φόρτου εντός των δικτύων και άρα αύξησης της απόδοσης του. Όταν σε ένα δίκτυο λειτουργεί και IP τηλεφωνία εντάσσεται σε διαφορετικό VLAN για λόγους διασφάλισης ποιότητας υπηρεσίας (QoS)
- Γίνεται καλύτερη ομαδοποίηση των σταθμών εργασίας ανάλογα με την ομάδα εργασίας που ανήκουν οι χρήστες τους
- Με τη χρήση λιγότερων μεταγωγέων που συνεπάγεται η σχεδίαση VLANs επιτυγχάνεται μείωση του φορτίου που απαιτεί το Spanning Tree Protocol (STP)
- Για λόγους ασφαλείας πολλές φορές χρειάζεται να εντάξουμε υπολογιστές που διαχειρίζονται ευαίσθητες πληροφορίες σε ξεχωριστό VLAN

VPNs: Η άλλη μέθοδος link virtualization αφορά την υλοποίηση ιδιωτικών εικονικών δικτύων (Virtual Privates Networks – VPNs) που είναι δίκτυα υπολογιστών που χρησιμοποιούν τις δημόσιες τηλεπικοινωνιακές υποδομές για να παρέχουν ασφαλή

πρόσβαση απομακρυσμένων χρηστών με εταιρικά δίκτυα. Σκοπός είναι η αποφυγή χρήσης των ακριβών ιδιωτικών ή μισθωμένων γραμμών που προορίζονται αυστηρά για μια εταιρεία. Ένα VPN δίκτυο αποτελείται από provider edge routers (PE) και customer edge συσκευές (CE). Τα VPNs ανάλογα με τα πρωτόκολλα που χρησιμοποιούν χωρίζονται στις εξής κατηγορίες:

- Layer 1 VPN: Επιτρέπει την μεταφορά πληροφορίας που έχει ενθυλακωθεί σε ATM ή IP πακέτα. Χρησιμοποιείται για να επεκτείνει τη μεταγωγή πακέτων των Layer 2 / Layer 3 VPNs σε συστήματα μεταγωγής κυκλώματος.
- Layer 2 VPN: Υποστηρίζει την μεταφορά πλαισίων Ethernet μεταξύ των δικτυακών τόπων που διασυνδέει. Το πλεονέκτημα αυτής της κατηγορίας ιδιωτικών εικονικών δικτύων είναι ότι αγνοούν τα πρωτόκολλα υψηλότερου επιπέδου προσφέροντας έτσι μεγάλη ευελιξία.
- Layer 3 VPN: Χρησιμοποιείται για την μεταφορά δεδομένων μεταξύ των PE routers στο δίκτυο κορμού του VPN και χωρίζεται σε δύο υποκατηγορίες: στα CE δίκτυα εκείνα που οι δικτυακοί τόποι συνδέονται χωρίς την μεσολάβηση δρομολογητών άμεσα μέσω ασφαλών συνδέσεων καθώς και στα PE δίκτυα που υπάρχει συγκεκριμένη διαχείριση κατά την λειτουργία τους.
- VPNs υψηλότερων επιπέδων: Εκτός από τις παραπάνω κατηγορίες VPN υπάρχουν και VPN που χρησιμοποιούν ανώτερα στρώματα της ιεραρχίας OSI όπως το στρώμα μεταφοράς ή το στρώμα εφαρμογής. Αυτά χρησιμοποιούνται για συγκεκριμένες εφαρμογές όπως για παράδειγμα τα SSL VPNs. Το SSL παρέχει κρυπτογράφηση μεταξύ των web servers που τρέχουν το SSL και των browsers. Άλλο παράδειγμα είναι το SSH, ένας μηχανισμός για ασφαλείς και κρυπτογραφημένες συνδέσεις σε διάφορες δικτυακές συσκευές. Μπορεί να δημιουργήσει VPNs για άλλα πρωτόκολλα του επιπέδου εφαρμογής όπως FTP και HTTP.

Programmable Networks [14]

Νέες δικτυακές δυνατότητες στα IP δίκτυα, όπως το QoS, απαιτούν την ανάπτυξη νέων δικτυακών υπηρεσιών και μάλιστα δυναμικά και σε πραγματικό χρόνο. Η έννοια των προγραμματιζόμενων δικτύων βασίζεται στο διαχωρισμό της υπολογισιμότητας και των δυνατοτήτων δρομολόγησης των δικτυακών κόμβων. Αυτό επιτυγχάνεται διαχωρίζοντας το τηλεπικοινωνιακό υλικό από το λογισμικό ελέγχου. Οι προγραμματιζόμενοι κόμβοι επιτρέπουν σε τρίτες οντότητες να τους προγραμματίζουν δυναμικά, ώστε να επεκτείνουν

τη λειτουργικότητά τους και να παρέχουν νέες υπηρεσίες. Τα προγραμματιζόμενα δίκτυα προκύπτουν όταν αυτές οι νέες υπηρεσίες υποστηρίζονται σε όλη την έκτασή τους από τους κόμβους. Έτσι αποτελούν ένα νέο μέσο ανάπτυξης γρήγορων και δυναμικών υπηρεσιών, εύκολης παραμετροποίησης και έλεγχου πόρων. Αφού επιτρέπεται η δυναμική αλλαγή του λογισμικού των μεταγωγέων και δρομολογητών, προκύπτουν νέα ζητήματα αναφορικά με την ασφάλεια, τον έλεγχο αποδοχής, τη διαχείριση των πόρων και τη συμβατότητα με παλαιότερα συστήματα. Η ασφάλεια επηρεάζεται από το πλήθος των πόρων στους οποίους μπορεί να έχει πρόσβαση μια εξωτερική οντότητα. Η διαχείριση των πόρων σχετίζεται με τη δικαιοσύνη και το κατά πόσο επηρεάζονται οι κόμβοι από τις αλλαγές στο λογισμικό τους. Η αλληλεπίδραση με παλαιά συστήματα αντιμετωπίζει το βαθμό προγραμματισιμότητας που μπορεί να υποστηριχθεί από μη-προγραμματιζόμενους κόμβους. Τα προγραμματιζόμενα δίκτυα χωρίζονται σε τρεις βασικές κατηγορίες: τα ενεργά δίκτυα (Active Networks), τις Open Programmable Interface Technologies και τα υβριδικά δίκτυα που περιλαμβάνουν και τις δυο προηγούμενες προσεγγίσεις.

Τα **ενεργά δίκτυα** υιοθετούν μια δυναμική ανάπτυξη νέων υπηρεσιών σε πραγματικό χρόνο. Τα πακέτα δεδομένων είναι υπεύθυνα για τη μεταφορά του κώδικα στους απομακρυσμένους κόμβους και αποτελούν τη βάση ελέγχου του δικτύου. Μεταφέρουν εκτελέσιμο κώδικα ή οδηγίες μέσω των οποίων ο κόμβος εκκινεί εφαρμογές ή υπηρεσίες που απαιτεί ο τελικός χρήστης. Όσο αφορά το Open Programmable Interface οι προγραμματιζόμενες διεπαφές είναι αρκετά καλά καθορισμένες μέσω κάποιου API. Ο συνδυασμός των δύο συνιστά ένα υβριδικό δίκτυο και η διαμόρφωση αυτή εξαρτάται από τον τύπο της εφαρμογής.

Η έννοια των ενεργών δικτύων εμφανίστηκε για πρώτη φορά στα μέσα της δεκαετίας του ’90 από την DARPA. Τα πρώτα προβλήματα που παρουσιάστηκαν και ώθησαν στην ανάπτυξη νέων τεχνολογιών στις υπάρχουσες τεχνολογίες Δικτύου ήταν:

- η δυσκολία στην ενσωμάτωση των νέων τεχνολογιών και πρωτοτύπων στην υποδομή των διαμοιραζόμενων δικτύων.
- η χαμηλή απόδοση κάποιων επιπέδων του πρωτοκόλλου του δικτύου στην προσπάθεια μείωσης των διαφόρων λειτουργιών.
- η δυσκολία προσαρμογής κάποιων νέων υπηρεσιών στο υπάρχον αρχιτεκτονικό μοντέλο.

Για τη επίλυση των προαναφερόμενων προβλημάτων εμφανίστηκε η προαναφερθείσα τεχνολογία των ενεργών δικτύων [12]. Η ιδέα ενός μηνύματος που θα μεταφέρει

διαδικασίες και δεδομένα είναι σίγουρα ένα βήμα πιο πάνω από τον συμβατικό τρόπο μεταγωγής κυκλώματος ή πακέτου και είναι το δικτυακό αντίστοιχο του αντικειμενοστραφούς προγραμματισμού, ο οποίος θεωρεί τα δεδομένα και τις διεργασίες που επιδρούν πάνω σε αυτά ως ένα αδιάσπαστο αντικείμενο.

Σε ένα περιβάλλον μπορούμε, εκτελώντας λειτουργίες εντός των κόμβων του δικτύου, να έχουμε μια προσέγγιση βασισμένη σε προγράμματα, η οποία μπορεί να χρησιμοποιηθεί για τη θεμελίωση ενός συστήματος δικτύου το οποίο θα είναι η σύνθεση πολλών μικρότερων στοιχείων με συγκεκριμένες ιδιότητες.

Οι υπηρεσίες μπορούν να κατανεμηθούν και να αναπαρασταθούν σύμφωνα με τις απαιτήσεις των εφαρμογών. Επίσης, η συνολική συμπεριφορά του δικτύου μπορεί να περιγραφεί από τις ιδιότητες των μεμονωμένων στοιχείων.

Για την πραγματοποίηση των ενεργών δικτύων υπάρχουν δύο διαφορετικές προσεγγίσεις. Η προσέγγιση του προγραμματιζόμενου μεταγωγέα (programmable switch) όπου τα πακέτα διατηρούν την τυποποίηση τους ενώ ταυτόχρονα παράγουν διακριτούς μηχανισμούς οι οποίοι υποστηρίζουν τη μεταφορά (download) προγραμμάτων. Ο διαχωρισμός της ενσωμάτωσης (injection) των προγραμμάτων από την διαδικασία εκτέλεσης προγραμμάτων των μηνυμάτων είναι ιδιαίτερα ελκυστικός όταν η επιλογή των προγραμμάτων γίνεται από τους διαχειριστές του δικτύου αντί των τελικών χρηστών.

Αντίθετα, η προσέγγιση κάψουλας (capsule) προχωρά ακόμη περισσότερο αφού τα «παθητικά» πακέτα των υπαρχόντων αρχιτεκτονικών αντικαθίστανται από ενεργά μικροπρογράμματα (miniature programs) τα οποία ενθυλακώνονται στα απεσταλμένα πλαίσια και εκτελούνται σε κάθε κόμβο κατά μήκος της διαδρομής τους. Αυτό το είδος ενεργών δικτύων αντιπροσωπεύει την πιο καινοτόμο από τις δυο ιδεές.

Η έρευνα στα ενεργά δίκτυα περιστρέφεται γύρω από δύο τεχνολογίες της ώθησης-έλξης (push-pull). Η ιδέα για την τεχνολογία έλξης έχει άμεση σχέση με τα firewalls, τους web proxies, τους multicast routers, τους mobile proxies, τα video gateways, όπου υλοποιούνται υπολογισμοί από τους χρήστες (user-driven computation) εντός των κόμβων του δικτύου.

Σε πολλές περιπτώσεις οι υπηρεσίες αυτές εκτελούνται στους κόμβους, όπως για παράδειγμα στην περίπτωση των firewalls τα οποία σχετίζονται άμεσα με τους δρομολογητές εκτελώντας ακόμη και συγκεκριμένες εφαρμογές, υπερβαίνοντας με αυτό τον τρόπο τις τυπικές αρχές της αρχιτεκτονικής των δικτύων.

Στόχος είναι η αντικατάσταση όλων των προσεγγίσεων που έχουν γίνει για συγκεκριμένους σκοπούς με ένα δίκτυο που θα δίνει την δυνατότητα στους χρήστες να το προγραμματίζουν σύμφωνα με τις ανάγκες τους [11].

Η τεχνολογία ώθησης κάνει εφικτούς τους στόχους μας με την εμφάνιση των ενεργών τεχνολογιών. Έως και πρόσφατα το να προγραμματίζουν οι διαχειριστές τα δίκτυα τους ήταν κάτι το αδιανόητο και αυτό γιατί σέβονταν την ασφάλεια και την αποτελεσματικότητα των υποδομών τους. Ωστόσο με τις σύγχρονες προηγμένες γλώσσες προγραμματισμού, τους μεταγλωττιστές και τα λειτουργικά συστήματα μπορούν να παραχθούν τα κλειδιά για ασφαλή και αποτελεσματική εκτέλεση κινούμενων κομματιών προγραμμάτων.

Αυτές οι ενεργές τεχνολογίες εφαρμόζονταν καταρχήν στα τερματικά συστήματα (end systems) έως και το επίπεδο δικτύου, για παράδειγμα η δυνατότητα ανταλλαγής Java applets μεταξύ των εξυπηρετητών (web servers) και πελατών (clients). Τα ενεργά δίκτυα ενισχύουν και επεκτείνουν αυτές τις τεχνολογίες για χρήση εντός του δικτύου με τρόπο τέτοιο ώστε να αλλάξει εκ θεμελίων η αντίληψη για το τι μπορεί να γίνει εντός του δικτύου.

Στα ενεργά δίκτυα, σε αντίθεση με τα κλασικά όπου οι δρομολογητές υλοποιούν σχεδόν ισοδύναμους υπολογισμούς σε κάθε πακέτο, όλοι οι κόμβοι μπορούν να υπολογίζουν ισοδύναμα υπολογιστικά μοντέλα. Οι ενεργοί κόμβοι μπορούν να εκτελούν πολλά διαφορετικά προγράμματα, για παράδειγμα μπορούν να υλοποιούν διαφορετικούς υπολογισμούς σε καθένα από τα πακέτα που τους διατρέχουν. Αυτό επιτυγχάνεται λειτουργώντας αφαιρετικά κατά τρόπο τέτοιο ώστε η διαλειτουργικότητα να εξασφαλίζεται επιτρέποντας στις εφαρμογές να βελτιστοποιούν την διαδικασία των μηνυμάτων σύμφωνα με τις εκάστοτε ανάγκες. Ένας τέτοιος μηχανισμός αφαίρεσης είναι τα virtual instruction sets.

Η δυνατότητα για φόρτωμα (download) νέων υπηρεσιών στις υποδομές θα μας οδηγήσει σε μια διαδικασία που θα κατευθύνεται από το χρήστη, όπου η διαθεσιμότητα των νέων υπηρεσιών θα εξαρτάται από την αποδοχή τους στην ανταγωνιστική αγορά. Τα ενεργά δίκτυα δίνουν τη δυνατότητα για αλλαγές στην βιομηχανία των δικτύων από ένα κεντρικό (mainframe) σκεπτόμενο σύνολο (mind-set), όπου το υλικό και το λογισμικό είναι άρρηκτα συνδεδεμένο, σε μια εικονική (virtualized) προσέγγιση όπου το λογισμικό είναι ανεξάρτητο του υλικού.

Τα ενεργά δίκτυα θα δώσουν την δυνατότητα να ξεκινήσουν νέες εφαρμογές που θα βασίζονται στη συγχώνευση της πληροφορίας στο δίκτυο (network-based merging of information), στην προστασία του δικτύου με τη συμμετοχή του χρήστη (user-aware network protection) και στην ενεργή διαχείριση του δικτύου (active network management).

Συγχώνευση και διαμοιρασμός πληροφορίας:

Χαρακτηριστικό δείγμα των πολυεφαρμογών, multi-user, multi-site, ήταν το Mbone. Υπάρχουν πάρα πολλές εφαρμογές οι οποίες απαιτούν υπηρεσίες βασιζόμενες στο δίκτυο που να υποστηρίζουν την συγχώνευση και το διαμοιρασμό της πληροφορίας. Ωστόσο υπάρχουν συστήματα που στηρίζονται σε υπηρεσίες που παρέχουν πολύ περιορισμένες λειτουργίες όπως για παράδειγμα η αντιγραφή των πακέτων IP οι οποίες δεν υποστηρίζουν την διανομή συγκεκριμένων εφαρμογών αφήνοντας έτσι χωρίς βοήθεια την αποθήκευση στο δίκτυο ή τη συγχώνευση της πληροφορίας. Οι Multi-site εφαρμογές θα αυξήσουν την υπολογιστική και αποθηκευτική δύναμη της πληροφορίας εντός του δικτύου. μια εφαρμογή όπως μια εξομοίωση ή εξ αποστάσεως διαχείριση επιτρέπει σε κάθε χρήστη να «δει» σύνθετες εικόνες δομημένες από τη συγχώνευση της πληροφορίας που προέρχεται από ένα μεγάλο αριθμό αισθητήρων. μια εφαρμογή όπως μια εξομοίωση ή εξ αποστάσεως διαχείριση επιτρέπει σε κάθε χρήστη να «δει» σύνθετες εικόνες δομημένες από τη συγχώνευση της πληροφορίας που προέρχεται από ένα μεγάλο αριθμό αισθητήρων.
Επιπλέον, τον κάθε αισθητήρα μπορεί να τον βλέπει ένας αριθμός χρηστών όπου ο καθένας από αυτούς θα έχει διαφορετικές απαιτήσεις σύμφωνα με τη κωδικοποίηση και παρουσίαση της πληροφορίας που προσπελαύνει. Η συγχώνευση των δεδομένων εντός του δικτύου μειώνει τις απαιτήσεις σε εύρος ζώνης (bandwidth) στον τελικό χρήστη. Ομοίως εξειδικευμένες σε κάθε χρήστη υπηρεσίες πολλαπλής μετάδοσης (user-specific multicast) εντός του δικτύου μειώνουν το φορτίο των αισθητήρων και του backbone του δικτύου.
Οι web proxies οι οποίοι αποθηκεύουν σελίδες πληροφορίας είναι ένα ακόμη παράδειγμα υπηρεσιών πολλαπλών χρηστών που μπορούν να αποκομίσουν οφέλη από υπολογισμούς και αποθήκευση δεδομένων εντός του δικτύου. Υπάρχουν κάποια ιεραρχικά μοντέλα προσωρινής αποθήκευσης όπως για παράδειγμα το Harvest, τα οποία μπορούν να μειώσουν τις καθυστερήσεις που οφείλονται στους χρήστες και να κανονικοποιήσουν το εύρος ζώνης που καταναλώνεται. Οι κόμβοι που χρησιμοποιούνται σαν προσωρινές μνήμες τοποθετούνται κοντά στα τερματικά σημεία του δικτύου, για παράδειγμα κόμβοι εντός των τελικών χρηστών που βρίσκονται σε κάποιο οργανισμό. Τα συστήματα αυτά μπορούν να επεκταθούν επιτρέποντας στους κόμβους που δομούνται ιεραρχικά, να τοποθετούνται σε σημεία στρατηγικής σημασίας εντός του δικτύου των παρόχων του Διαδικτύου και να μοιράζονται τους φορείς. Ένα αρκετά ενδιαφέρον πρόβλημα είναι η ανάπτυξη αλγορίθμων και εργαλείων τα οποία θα εξισορροπούν την ιεραρχική δομή με την ενημέρωση των προσωρινών μνημών και όχι μόνο της αποθηκευμένης πληροφορίας. Ένα επιπλέον επιχείρημα για την χρήση των ενεργών τεχνολογιών για το Web caching είναι ότι σε ένα

σημαντικό ποσοστό ιστοσελίδων οι υπολογισμοί γίνονται δυναμικά. Για το λόγο αυτό προτείνεται η ανάπτυξη σχεδίων τέτοιων, που να υποστηρίζουν ενεργές προσωρινές μνήμες οι οποίες θα αποθηκεύουν και θα εκτελούν τα προγράμματα «γεννήτορες» αυτών των σελίδων.

User-aware network protection:

Ο όρος ασφάλεια της πληροφορίας συνδέεται στενά με τον όρο προστασία της πληροφορίας με τον οποίο εννοούμε ότι η σωστή πληροφορία φτάνει στο σωστό άνθρωπο την κατάλληλη στιγμή και στον ζητούμενο τόπο. Το ζήτημα της ασφάλειας του δικτύου καθώς και της εύρεσης μηχανισμών πιστοποίησης είναι σίγουρα μείζον και έχουν δαπανηθεί πάρα πολλές ανθρωποώρες για την επίλυσή του. Παρόλα αυτά στα ενεργά δίκτυα είναι αποδεκτός ο σχεδιασμός ενός ολοκληρωμένου μηχανισμού όπου θα γίνεται διαχείριση όλων των πηγών του δικτύου καθώς και της πληροφορίας που το διατρέχει. Με τον σχεδιασμό αυτό εκλείπει η ανάγκη ενός συστήματος πολλαπλής ασφάλειας και πιστοποίησης το οποίο θα λειτουργεί ανεξάρτητα για κάθε επίπεδο του πρωτοκόλλου επικοινωνίας. Αυτό μας δίνει τη δυνατότητα να προγραμματίζουμε με ασφάλεια έχοντας σαν βάση το χρήστη ή τη χρήση. Τέλος μια τυπική προσέγγιση με τη χρήση αυστηρών προδιαγραφών και ασφάλεια τύπων επιβαλλομένης από τη γλώσσα (language enforced type-safety) μπορεί να χρησιμοποιηθεί ώστε να υλοποιούνται πολιτικές ασφάλειας.

Active Network Management:

Πολλές από τις αρμοδιότητες της διαχείρισης του δικτύου συνίστανται στη συλλογή και την αντιπαραβολή των δεδομένων κατά την αποτίμηση ενός γεγονότος. Οι ενεργές τεχνολογίες μπορεί να χρησιμοποιηθούν για να εφαρμοστεί μια πολύπλοκη προσέγγιση για την παρακολούθηση του δικτύου και το φιλτράρισμα των συμβάντων. Τα στοιχεία του δικτύου, όπως οι δρομολογητές, μπορούν να αναλάβουν ως ένα βαθμό την παρακολούθησή τους, ενσωματώνοντας διαγνωστικά προγράμματα στους κοντινότερους γείτονες τους. Με παρόμοιο τρόπο τα ενεργά δίκτυα μπορούν να παρέχουν την απαιτούμενη ευκαμψία που θα τους δίνει τη δυνατότητα να ανιχνεύουν λάθη και να ενημερώνουν τις πολιτικές επιβίωσης που ορίζει το σύστημα σε ακραίες καταστάσεις καταστροφών [13].

Overlay Networks:

Ένα overlay network είναι ένα λογικό δίκτυο που κατασκευάζεται πάνω από υπάρχουσες δικτυακές υποδομές. Το διαδίκτυο από μόνο του είναι ένα τέτοιο «δίκτυο» αφού ξεκίνησε βασίζοντας την λειτουργία του στις τηλεπικοινωνιακές γραμμές PSTN. Επίσης, τα εταιρικά ιδιωτικά δίκτυα που βασίστηκαν είτε στο Frame Relay είτε στο ATM αποτελούν χαρακτηριστικό παράδειγμα overlay networks. Πλέον τα προαναφερθέντα δίκτυα έχουν αντικατασταθεί από τα MPLS δίκτυα και VPNs. Τα υπάρχοντα overlays δίκτυα υλοποιούνται κυρίως στο επίπεδο εφαρμογής του OSI. Υπάρχουν υλοποιήσεις overlay δικτύων και σε χαμηλότερα επίπεδα. Τα overlay δίκτυα δεν απαιτούν κάποια αλλαγή στα υπάρχοντα δίκτυα πάνω στα οποία τρέχουν. Τα δίκτυα αυτά έχει επιχειρηθεί να αξιοποιηθούν είτε για βελτίωση της επίδοσης είτε για την υποστήριξη πολυεκπομπής (multicasting) με σκοπό την βελτίωση της ποιότητας υπηρεσίας. Έχουν χρησιμοποιηθεί και ως πειραματικές πλατφόρμες (testbeds) όπως το PlanetLab με στόχο την αποτίμηση νέων αρχιτεκτονικών.

2.2 Πλεονεκτήματα εικονικοποίησης συστήματος

Τα πλεονεκτήματα που απορρέουν από την εφαρμογή της εικονικοποίησης σε έναν κόμβο ενός δικτύου είναι τα ακόλουθα:

- Δυνατότητα ταυτόχρονης εκτέλεσης πολλών λειτουργικών συστημάτων σε έναν υπολογιστή. Μας επιτρέπεται έτσι να εκτελούμε σε ξεχωριστές virtual machine εφαρμογές που έχουν διαφορετικές απαιτήσεις όσο αφορά το λειτουργικό σύστημα.
- Δυνατότητα εκτέλεσης μιας εφαρμογής που ήδη εκτελείται σε έναν server σε διαφορετικό server (migration) χωρίς downtime, διασφαλίζοντας έτσι υψηλή διαθεσιμότητα της εφαρμογής. Η δυνατότητα αυτή είναι χρήσιμη σε περιπτώσεις κατάρρευσης ενός server ή για την συντήρηση του.
- Περιορισμός του κόστους λειτουργίας και συντήρησης ενός datacenter διότι η εφαρμογή της εικονικοποίησης οδηγεί σε ελάττωση του απαιτούμενου υλικού

- Περιορίζεται η κατανάλωση ενέργειας λόγω της μείωσης του απαιτούμενου hardware (Green IT). Με το virtualization ο διαχειριστής συστήματος μπορεί να συνδυάσει πολλά φυσικά συστήματα σε virtual machines που τρέχουν σε ένα σύστημα, μειώνοντας έτσι την κατανάλωση σε ενέργεια και σε απαιτούμενη ψύξη. Επίσης γίνεται καταμερισμός φόρτου εργασιών στους servers, γεγονός που επίσης συντελεί σε μειωμένη κατανάλωση
- Μπορούμε να εκτελούμε στον ίδιο εξυπηρετητή ταυτόχρονα πληθώρα εφαρμογών διασφαλίζοντας έτσι υψηλή χρησιμοποίηση των πόρων. Στους σύγχρονους εξυπηρετητές έχουμε χρησιμοποίηση γύρω στο 5-15% των πόρων. Η εφαρμογή της εικονικοποίησης μας επιτρέπει να αξιοποιήσουμε καλύτερα τους διαθέσιμους πόρους.
- Απλοποιημένη διαχείριση από απόσταση αφού αρκεί μόνο μια απλή σύνδεση στο διαδίκτυο. Με τα κατάλληλα εργαλεία διαχείρισης του συνόλου των πόρων που προκύπτουν στο εικονικό περιβάλλον μπορούμε να διαχειριστούμε με δυναμικό τρόπο τους πόρους σε επίπεδο υλικού καθώς κάθε εφαρμογή χαρακτηρίζεται από διαφορετικές απαιτήσεις σε επίπεδο πόρων.
- Δυνατότητα χρησιμοποίησης Thin Clients. Οι Thin Clients είναι τερματικά χαμηλού κόστους χωρίς περιφερειακά, συσκευές εισόδου/εξόδου και λειτουργούν σχεδόν αποκλειστικά με δικτυακό τρόπο και απαιτούν μηδαμινή συντήρηση.
- Δυνατότητα χρήσης υπάρχοντων τερματικών για απομακρυσμένη σύνδεση σε virtual machines που βρίσκονται στον κεντρικό εξυπηρετητή.

2.3 Περιγραφή εικονικού δικτυακού περιβάλλοντος

2.3.1 Διαχωρισμός ρόλων ISPs

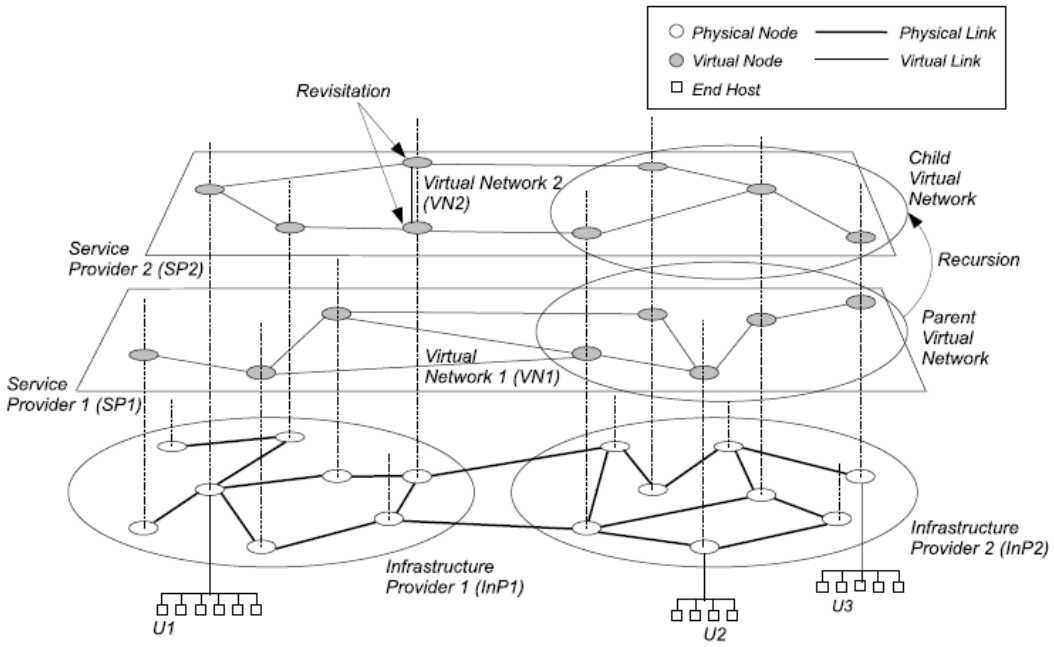
Οι βασικοί συντελεστές στο υπάρχον διαδίκτυο είναι οι πάροχοι υπηρεσιών πχ Google και οι internet services providers (ISPs). Ένας ISP προσφέρει στους πελάτες του υπηρεσίες πρόσβασης στο διαδίκτυο είτε μέσω της δικής του υποδομής είτε μισθώνοντας υποδομή από άλλους ISPs είτε ως συνδυασμό των δυο παραπάνω. Υπάρχουν τρεις διακριτοί ρόλοι που κυριαρχούν στο σημερινό διαδίκτυο και αφορούν τους παρόχους υπηρεσιών: ο φυσικός παροχέας υποδομής – Physical Infrastructure provider (PIP) που κατέχει και διαχειρίζεται μια φυσική υποδομή, ο connectivity provider που παρέχει bit-pipes και από άκρη σε άκρη επικοινωνία με τους τελικούς χρήστες, και ο service provider που προσφέρει εφαρμογές, δεδομένα και υπηρεσίες περιεχομένου στους τελικούς χρήστες. Με την εισαγωγή τεχνικών εικονικοποίησης οι διακριτοί ρόλοι συνοψίζονται σε τέσσερις και είναι οι εξής:

- Physical Infrastructure providers που κατέχουν και διαχειρίζονται την φυσική υποδομή
- Virtual Network Providers (VNP) που είναι υπεύθυνοι για την συνένωση των εικονικών πόρων από πολλαπλούς PIPs και την δημιουργία ενιαίων εικονικών τοπολογιών
- Virtual Network Operators (VNO) που είναι υπεύθυνοι για την εγκατάσταση και διαχείριση ενός εικονικού δικτύου που παρέχεται από τον VNP σύμφωνα με τις ανάγκες του παρόχου υπηρεσιών
- Service Provider (SP) που χρησιμοποιεί το εικονικό δίκτυο που του προσφέρεται στοχεύοντας στην προσφορά υπηρεσιών προστιθέμενης αξίας ενεργώντας είτε ως application service provider είτε ως network service provider στην περίπτωση που προσφέρονται υπηρεσίες μεταφοράς δεδομένων.

Συνοψίζοντας, η εισαγωγή της εικονικοποίησης ως θεμελιώδους τεχνολογίας πάνω στην οποία θα βασιστεί η ανάπτυξη του μελλοντικού διαδικτύου [1] ουσιαστικά θέτει το θέμα του διαχωρισμού των ρόλων των υπαρχόντων παρόχων υπηρεσιών διαδικτύου (Internet Service Providers - ISPs). Ο ένας ρόλος που σχετίζεται με τη διαχείριση των πραγματικών υποδομών θα αφορά τους παρόχους υποδομής PIPs και ο άλλος ρόλος που σχετίζεται με την παροχή των υπηρεσιών θα αφορά τους παρόχους υπηρεσιών (Service Providers - SPs) που θα υλοποιούν υπηρεσίες χρησιμοποιώντας εικονικές υποδομές που έχουν αναπτύξει οι VNOs σε συνεργασία με τους VNPs.

2.3.2 Εικονικά δίκτυα και κόμβοι

Βασικό στοιχείο ενός εικονικοποιημένου περιβάλλοντος είναι τα εικονικά δίκτυα (Virtual Networks). Ένα εικονικό δίκτυο είναι ένα σύνολο από εικονικούς κόμβους συνδεδεμένους μεταξύ τους με ένα σύνολο από εικονικές ζεύξεις έτσι ώστε να σχηματίζουν μια εικονική τοπολογία που είναι μέρος της υπάρχουσας τοπολογίας υποδομής. Κάθε εικονικός κόμβος είναι ένα εικονικό μηχάνημα (virtual machine) που τρέχει σε έναν κόμβο υποδομής και διαθέτει τις δικές του αυτόνομες εικονικές διεπαφές (virtual interfaces – vNICS) που του επιτρέπουν να διασυνδεθεί μέσω εικονικών ζεύξεων με άλλους εικονικούς κόμβους. Ο εικονικός κόμβος αποτελείται από εικονικές μηχανές που είναι εγκατεστημένες σε αυτόν και κάθε μια λειτουργεί ανεξάρτητα από την άλλη κατά τρόπο που θα εξηγηθεί καλύτερα παρακάτω. Κάθε εικονικό δίκτυο ελέγχεται από έναν αποκλειστικό πάροχο υπηρεσίας ενώ η υποδομή του μπορεί να μισθωθεί από πολλούς παρόχους υποδομής (InPs).



2.3.3 Εικονικά Μηχανήματα

Ένα εικονικό μηχάνημα [3], [8] (virtual machine – VM) είναι μια software υλοποίηση ενός υπολογιστή που δύναται να εκτελεί προγράμματα ως πραγματικός υπολογιστής. Διαφέρει από ένα process VM που σχεδιάζεται για να τρέξει ένα μόνο πρόγραμμα όπως στην περίπτωση του Java Runtime Environment. Ένα system VM μπορεί να υποστηρίξει την εκτέλεση ενός πλήρους λειτουργικού συστήματος (OS). Ουσιαστικά τα εικονικά μηχανήματα προσομοιώνουν το υλικό ενός υπολογιστή και στη ουσία είναι υπολογιστές μέσα σε υπολογιστές. Οι πόροι ενός φυσικού συστήματος μοιράζονται στα διάφορα εικονικά μηχανήματα που τρέχουν σε αυτό και τα οποία διαθέτουν όλα τα χαρακτηριστικά γνωρίσματα ενός υπολογιστή, αποτελώντας έτσι αντίγραφα του hardware του υπολογιστή. Σε αυτά τα εικονικά μηχανήματα μπορεί να εγκατασταθεί οποιοδήποτε σχεδόν λειτουργικό σύστημα, μπορούν επίσης να υποστηρίξουν point to point συνδέσεις με άλλα εικονικά μηχανήματα χάρις στις εικονικές διεπαφές (vNICs) που διαθέτουν. Μέσω της διασύνδεσης των εικονικών μηχανημάτων διάφορων φυσικών κόμβων μεταξύ τους μέσω εικονικών ζεύξεων έχουμε τη δημιουργία **slices**. Ο κύκλος ζωής ενός εικονικού μηχανήματος αποτελείται από 6 φάσεις: create, suspend, resume, save, migrate και destroy. Σε έναν φυσικό κόμβο μπορούν να τρέχουν πολλαπλά εικονικά μηχανήματα και σε κάθε ένα από αυτά μπορεί να εκτελείται ένα λειτουργικό σύστημα. Σε κάθε κόμβο, όταν αναφερόμαστε

σε εικονικοποίηση εξυπηρετητή, υπάρχει ένα Virtual Machine Monitor (VMM) με σκοπό τη διαχείριση και τον έλεγχο των εικονικών μηχανήματων σε αυτόν τον κόμβο. Το VMM είναι εγκατεστήμενο στο hostOS του κόμβου, ενώ τα GuestOs είναι εγκατεστημένα εντός των virtual machines. Ένας VMM αναφέρεται πολλές φορές και ως hypervisor. Στην ουσία το VMM είναι ένα στρώμα λογισμικού ανάμεσα στο GuestOS και το hardware του κόμβου. Σε υψηλότερο ιεραρχικά επίπεδο βρίσκονται οι Virtual Infrastructure Managers με στόχο την διαχείριση και την αξιοποίηση και το monitoring σε μια κατανεμημένη δεξαμενή πόρων που μπορεί να είναι ένα cluster ή ένα datacenter.

Τα δύο βασικά σημεία που δίδεται έμφαση όσο αφορά την χρήση του virtualization για την ανάπτυξη υποδομών για το Future internet είναι:

- Οι συνδέσεις point to point είτε με εξασφαλισμένη χωρητικότητα και στρώμα ζεύξης δεδομένων (data link layer) είτε χωρίς
- Το εικονικό μηχάνημα (virtual machine) που αποτελείται από τη δική του μνήμη RAM, CPU, αποθηκευτικό χώρο κλπ όπως δηλαδή και ένας κανονικός υπολογιστής και υποστηρίζει την εκτέλεση διάφορων λειτουργικών συστημάτων και λειτουργιών όπως πχ η δρομολόγηση.

Ένα σημαντικό ζήτημα που προκύπτει με τους virtual πόρους είναι κατά πόσο συμπεριφέρονται όπως ακριβώς οι αντίστοιχοι πραγματικοί. Για παράδειγμα μια εικονική ζεύξη μπορεί να μην είναι σε θέση να προσφέρει σταθερό ρυθμό bit/sec.

2.3.4 Ιδιότητες εικονικού δικτυακού περιβάλλοντος

Συνύπαρξη (coexistence): Η συνύπαρξη πολλών VNs από πολλούς παρόχους συνίσταται στο γεγονός ότι μπορούν να συνυπάρχουν περισσότερες από μία φυσικές ζεύξεις σε έναν ή περισσότερους παρόχους υποδομών. Αυτό σημαίνει ότι ένας πάροχος υποδομής (Infrastucture Provider) θα πρέπει να είναι σε θέση να δώσει τους πόρους του σε πολλούς παρόχους υπηρεσιών οι οποίοι από την πλευρά τους μπορούν να δημιουργήσουν τα εικονικά δίκτυα τους δεσμεύοντας πόρους από πολλούς παρόχους υποδομής. Για παράδειγμα τα VN1 και VN2 είναι δύο συνυπάρχοντα δίκτυα.

Ενθυλάκωση εικονικών δικτύων (recursion): Σε κάποιες περιπτώσεις χρειάζεται η δημιουργία εικονικών δικτύων εντός άλλων εικονικών δικτύων οδηγώντας έτσι στη δημιουργία μιας ιεραρχίας σχέσεων του τύπου πατέρας – παιδί. Η σχέση αυτή είναι γνωστή

ως recursion και πρόκειται πρακτικά για φώλιασμα των εικονικών δικτύων.

Κληρονομικότητα (inheritance): Τα εικονικά δίκτυα - παιδιά κληρονομούν αρχιτεκτονικές των δικτύων γονέων που συνεπάγεται ότι οι περιορισμοί που υφίστανται στα εικονικά δίκτυα γονείς υφίστανται και στα παιδιά. Για παράδειγμα περιορισμοί που υφίσταται στο δίκτυο του InP2 θα υπάρχουν στο SP1 και κατά συνέπεια και στο SP2.

Επανεπισκεψιμότητα (Revisitation): Επιτρέπει σε έναν κόμβο υποδομής την φιλοξενία περισσότερων του ενός εικονικών κόμβων ενός μόνο εικονικού δικτύου. Η δομή του εικονικού δικτύου καθώς και η διαχείριση του γίνεται με χρήση πολλαπλών routers με λογικές συνδέσεις. Η ιδιότητα αυτή κρίνεται χρήσιμη για την δημιουργία πειραματικών δικτύων (testbeds). Στο παραπάνω σχήμα φαίνονται δύο εικονικοί κόμβοι του δικτύου SP2 να υφίσταται στον ίδιο κόμβο υποδομής.

2.3.5 Στόχοι εικονικού δικτυακού περιβάλλοντος

Ευελιξία (Flexibility): Κάθε SP πρέπει να είναι σε θέση να υλοποιεί οποιαδήποτε δικτυακή τοπολογία, καθώς επίσης και λειτουργίες δρομολόγησης και προώθησης ανεξάρτητα από τα υποκείμενα δίκτυα υποδομής και άλλα συνυπάρχοντα εικονικά δίκτυα.

Διαχειρισιμότητα (manageability): Με τον διαχωρισμό των ρόλων μεταξύ SPs και InPs η εικονικοποίηση προσφέρει τη δυνατότητα για διαχείριση από άκρη σε άκρη κάθε στρώματος του δικτύου. Η εικονικοποίηση θα πρέπει να παρέχει την δυνατότητα πλήρους ελέγχου του εικονικού δικτύου κάθε παρόχου υπηρεσιών.

Κλιμακωσιμότητα (Scalability): Η συνύπαρξη πολλών εικονικών δικτύων είναι μια θεμελιώδης αρχή της εικονικοποίησης των δικτύων. Οι πάροχοι υποδομής σε ένα δικτυακό εικονικοποιημένο περιβάλλον (NVE) πρέπει να αυξάνουν την χωρητικότητα τους ώστε να υποστηρίζουν το διαρκώς αυξανόμενο πλήθος εικονικών δικτύων χωρίς να επηρεάζουν την απόδοση τους.

Απομόνωση (isolation): Θα πρέπει να διασφαλίζεται απομόνωση μεταξύ των συνυπαρχόντων εικονικών δικτύων για λόγους ασφαλείας, ιδιωτικότητας καθώς και ανοχής

στα λάθη. Οποιαδήποτε δυσλειτουργία λαμβάνει χώρα σε ένα εικονικό δίκτυο πρέπει να γίνεται αποκλειστικά εντός αυτού και να μην επιδρά στα συνυπάρχοντα δίκτυα

Ευστάθεια και σύγκλιση (stability and convergence): Μολονότι η απομόνωση διασφαλίζει ότι μια δυσλειτουργία σε ένα εικονικό δίκτυο δεν θα επηρεάσει τα άλλα εικονικά δίκτυα που μοιράζονται την ίδια φυσική υποδομή υπάρχει περίπτωση όταν εμφανιστεί μια δυσλειτουργία να αποσταθεροποιηθεί το δίκτυο. Έτσι θα πρέπει να διασφαλίζεται ότι σε κάθε περίπτωση η ευστάθεια του εικονικοποιημένου περιβάλλοντος καθώς και ότι κάθε εικονικό δίκτυο θα πρέπει σε περίπτωση αστάθειας να συγκλίνει έπειτα στην ευσταθή κατάσταση του.

Προγραμματισιμότητα (Programmability): Διασφαλίζει ότι σε ένα εικονικό δικτυακό περιβάλλον (Network Virtual Environment - NVE) υπάρχει ευελιξία και διαχειρισιμότητα. Μόνο μέσω προγραμματισιμότητας ένας πάροχος μπορεί να υλοποιήσει πρωτόκολλα και να αξιοποιήσει ποικίλες υπηρεσίες. Δύο βασικά ζητήματα που τίθεται σχετικά με την προγραμματισιμότητα αφορούν τον βαθμό προγραμματισιμότητας ενός δικτύου και πως αυτή υλοποιείται.

Ετερογένεια: Αναφέρεται αφενός μεν στην ύπαρξη των πολλών ετερογενών δικτύων που απαρτίζουν την φυσική υποδομή πάνω στην οποία υλοποιείται η εικονικοποίηση (οπτικά, ασύρματα κλπ) και αφετέρου σε κάθε εικονικό δίκτυο που υλοποιείται πάνω στην ετερογενή υπάρχουσα υποδομή και μπορεί να είναι και αυτό ετερογενές. Κατά συνέπεια θα πρέπει τα δίκτυα υποδομής να υποστηρίζουν την δημιουργία πάνω σε αυτά κάθε είδους εικονικών δικτύων ανεξάρτητα από την τεχνολογία στην οποία βασίζονται και επιπλέον ετερογενή πρωτόκολλα και αλγορίθμους από πολλαπλούς παρόχους υπηρεσιών. Επιπλέον θα πρέπει να εκτιμηθεί και η ετερογένεια μεταξύ των τελικών συσκευών.

Legacy Support: Όταν υιοθετείται οποιαδήποτε νέα τεχνολογία, σοβαρό ρόλο παίζει η υποστήριξη παρωχημένου υλικού. Καθίσταται απαραίτητο να παρέχεται συμβατότητα μεταξύ παλαιάς και νέας τεχνολογίας. Το εικονικό δικτυακό περιβάλλον με χρήση εικονικών μηχανημάτων και εικονικών ζεύξεων που τα διασυνδέουν εξασφαλίζει απομόνωση των διάφορων τεχνολογιών που ενσωματώνονται στο εσωτερικό τους εξασφαλίζοντας έτσι την εκτέλεση νέας και παλιάς τεχνολογίας σε ένα σύστημα.

2.3.6 Διαχείριση των Virtualised Networks

Για την παροχή πρόσβασης στους εικονικούς πόρους για το λογισμικό από τη μεριά του client θα πρέπει να καθοριστεί αυστηρά η διεπαφή διαχείρισης. Αρκετές βασικές υπηρεσίες που έχουν προταθεί, ακολουθούν τις προτάσεις του DTMF (Distributed Management Task Force) [18]. Αυτές περιλαμβάνουν λειτουργίες εκκίνησης, παύσης, διακοπής, επανεκκίνησης και μετακίνησης ενός εικονικού μηχανήματος.

Είναι σημαντικό οι λειτουργίες να περιλαμβάνουν την παρακολούθηση των εικονικών και φυσικών πόρων. Αυτό είναι απαραίτητο για να είναι εφικτή η σωστή αντιμετώπιση όταν δεν είναι επαρκείς οι διαθέσιμοι πόροι, με σκοπό να καλυφθούν απαιτήσεις που δεν είχαν αρχικά προβλεφθεί. Οι παράμετροι που παρακολουθούνται είναι η διαθέσιμη μνήμη, διαθέσιμη υπολογιστική ισχύς και η διαθέσιμη χωρητικότητα δίσκων. Παρόλα αυτά, για να εξασφαλισθεί η αφαίρεση και η ευελιξία που επιτυγχάνεται με την εικονικοποίηση, οι διεπαφές διαχείρισης πρέπει να επιτρέπουν απευθείας πρόσβαση στα δεδομένα παρακολούθησης του φυσικού υλικού. Αντίθετα, το λογισμικό διαχείρισης πρέπει να αλλάζει τις παραμέτρους παρακολούθησης, υπολογίζοντας ταυτόχρονα οποιαδήποτε επιβάρυνση παράγεται από την συγκεκριμένη τεχνική εικονικοποίησης που έχει υλοποιηθεί. Αυτό εξασφαλίζει ότι το client λογισμικό δεν έχει γνώση της πραγματικής εικονικοποίησης που έχει υιοθετηθεί και ασχολείται μόνο με τους πόρους.

Πιο πολύπλοκες, υψηλού επιπέδου μέθοδοι μπορούν να σχεδιασθούν ομαδοποιώντας διάφορες βασικές υπηρεσίες. Το λογισμικό από τη μεριά του client δεν χρειάζεται έτσι να γνωρίζει τη διεπαφή διαχείρισης στο σύνολό της. Αντίθετα επικεντρώνεται σε συγκεκριμένες εργασίες όπως λειτουργίες διαχείρισης απόδοσης, σφαλμάτων ή τοπολογίας, αδιαφορώντας για διεργασίες που δεν αφορούν την υπο παρακολούθηση λειτουργία.

Η διεπαφή θα πρέπει επίσης να παρέχει έναν τρόπο ώστε οι υπηρεσίες client να συμβαδίζουν με τις απαιτήσεις QoS (Quality of Service). Για να γίνει αυτό είναι απαραίτητο να ανατίθενται εξασφαλισμένοι πόροι, επιτρέποντας με ιεραρχικό τρόπο πρόσβαση σε εγγυημένο διαμοιρασμό των διαθέσιμων δικτυακών υπηρεσιών. Η προκύπτουσα ιεραρχία διαφορετικών προτεραιοτήτων θα πρέπει να βρίσκεται σε συμφωνία με τις υπηρεσίες. Αυτό επηρεάζει επίσης την ανάκαμψη σε περίπτωση αποτυχίας, για παράδειγμα σε ένα περιβάλλον με συγκεκριμένο QoS, μια ξαφνική αποτυχία σε έναν εικονικό δρομολογητή δεν μπορεί να είναι αποδεκτή. Τέτοιες περιπτώσεις απαιτούν ιδιαίτερες στρατηγικές ανάκαμψης.

2.3.7 Διευθυνσιοδότηση σε slice-based αρχιτεκτονικές

Όπως προαναφέρθηκε, στο slice computing κάθε χρήστης χρησιμοποιεί ένα μέρος των πόρων ενός συστήματος (sliver) οργανώνοντας τα δικά του εικονικά δίκτυα με αντίστοιχα slivers άλλων φυσικών δικτυακών κόμβων. Τα slivers προκύπτουν είτε με της εικονικοποίησης των πόρων του component, με την δημιουργία δηλαδή εικονικών αντιγράφων, είτε μέσω κατάτμησης του component σε διακριτά σύνολα πόρων. Μεταξύ των διάφορων slices που οργανώνονται μέσω της διασύνδεσης των slivers είναι απαραίτητη η απομόνωση των πόρων προκειμένου να διασφαλίζεται ότι εφαρμογές που εκτελούνται σε ένα sliver ενός φυσικού κόμβου δεν επηρεάζουν κατά οποιοδήποτε τρόπο την εκτέλεση άλλων εφαρμογών σε άλλα sliver εντός του ίδιου φυσικού κόμβου. Η απομόνωση των πόρων αφορά το namespace, το security context και το performance context. Δεδομένου ότι τα σύγχρονα λειτουργικά συστήματα διαθέτουν μηχανισμούς για τον έλεγχο της άδειας πρόσβασης εξασφαλίζοντας έτσι security context καθώς επίσης διαθέτουν και μηχανισμούς για τον καταμερισμό των πόρων του συστήματος στους διάφορους χρήστες αναγκαία είναι η ανάπτυξη ενός μηχανισμού που θα εξασφαλίζει απομόνωση των διευθύνσεων. Η απομόνωση των διευθύνσεων μεταξύ των διάφορων slivers ενός κόμβου είναι επιθυμητή ώστε να εξασφαλιστεί ότι κάθε χρήστης ενός κόμβου που καταλαμβάνει το δικό του sliver έχει πρόσβαση σε συγκεκριμένα interfaces ή IP διευθύνσεις ενός κόμβου. Η φιλοσοφία πάνω στην οποία θα βασίζεται η απομόνωση των διευθύνσεων βασίζεται στην ανάθεση σε κάθε χρήστη (user_id) ενός λειτουργικού συστήματος μιας IP διεύθυνσης και ενός slice identifier (slice_id). Με αυτό τον τρόπο κάθε sliver ενός slice που φιλοξενεί έναν χρήστη διαθέτει τη δική του IP η οποία είναι προσβάσιμη μόνο από το slice που εντάσσεται το συγκεκριμένο sliver και όχι από τα άλλα συνυπάρχοντα slices. Η συγκεκριμένη προσέγγιση ορίζεται ως διευθυνσιοδότηση προσανατολισμένη στο χρήστη (User Oriented Adressing – UOA), σε αντίθεση με την παραδοσιακή αρχιτεκτονική του διαδικτύου όπου ανατίθεται IP διευθύνσεις στα interfaces ενός δικτυακού κόμβου και αυτές τις μοιράζονται οι διάφοροι χρήστες. Η UOA αρχιτεκτονική διακρίνει τις διευθύνσεις της σε δύο κατηγορίες, τις slice-wise διευθύνσεις και τις node-wise.

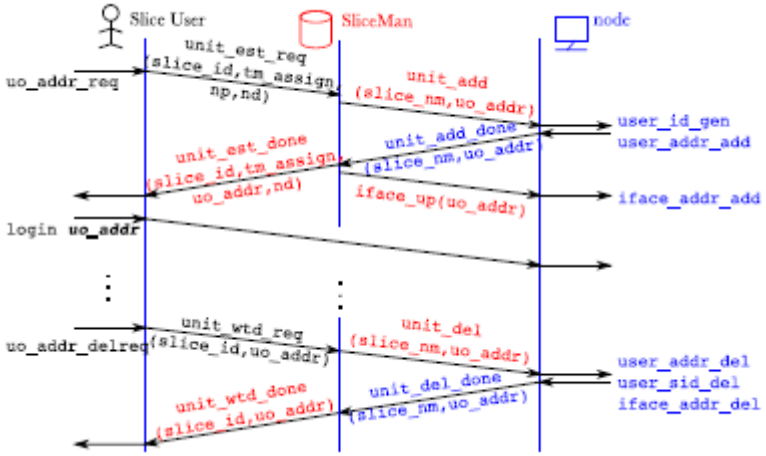
Slice-wise διευθύνσεις: Αναφέρονται όπως περιγράφηκε και παραπάνω σε IP διευθύνσεις που αποδίδονται ως αναγνωριστικά στους διάφορους χρήστες ενός κόμβου οι οποίοι είναι οργανωμένοι σε ξεχωριστά slivers κατά τα πρότυπα της container-based εικονικοποίησης, τα οποία συνδέονται με άλλα συνθέτοντας slices.

Node-wise διευθύνσεις: Στην περίπτωση αυτή τα slivers χρησιμοποιούνται για την εκτέλεση αποκλειστικά υπηρεσιών για τον κόμβο στον οποίο βρίσκονται. Κατά συνέπεια, δεν διασυνδέονται με άλλα slivers γειτονικών κόμβων και έτσι η IP διεύθυνση που τους αποδίδεται έχει καθαρά τοπικό χαρακτήρα εντός του κόμβου στον οποίο βρίσκονται. Η διαχείριση των node-wise διευθύνσεων γίνεται από τους διαχειριστές των κόμβων. Σε κάθε κόμβο οι διάφοροι χρήστες που φέρουν ο καθένας από ένα user_id αντιστοιχίζονται σε διαφορετικά slices, που το καθένα διαθέτει ένα συγκεκριμένο slice_id. Η διεύθυνση (uo_addrr) που προκύπτει με χρήση της UOA μεθοδολογίας είναι συνάρτηση ενός δικτυακού προθέματος – network prefix (n,p), του slice_id, καθώς και της χρονικής στιγμής που γίνεται η ανάθεση – time of assignment (t_a). Οι μετρητές χρόνου των διάφορων slices δεν είναι απαραίτητο να είναι συγχρονισμένοι. Είναι πιθανό ένα user_id να διαθέτει περισσότερες από μια uo_adrr παραγόμενες από το ίδιο slice_id με το ίδιο t_a αλλά διαφορετικό np. Μια UOA διεύθυνση πρέπει να προκύπτει γενικά ως συνδυασμός του δικτυακού προθέματος (np) και ενός user-oriented suffix που ορίζεται ως εξής

$$pseudo\_iface\_id = \phi_1(slice\_id) + \phi_2(t\_a)$$

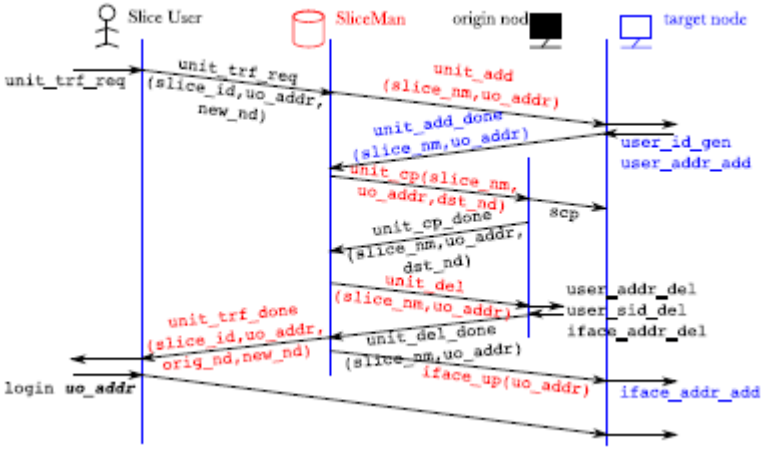
Η διαχείριση των UOA διευθύνσεων αναφέρεται στην ανάθεση, παραμετροποίηση, την μεταφορά τους σε περίπτωση μεταφοράς ενός slice (migration) από ένα σύνολο κόμβων σε ένα άλλο, καθώς και την απόσυρση τους. Η διαχείριση των διευθύνσεων γίνεται μέσω συγκεκριμένων πρωτοκόλλων με συντονισμό από τον SliceMan. Το SliceMan είναι ένας διαχειριστής των slices και διαθέτει μια βάση δεδομένων συσχετίζοντας τους χρήστες (user_id) με τα slices (slice_id) στα οποία υπάγονται καθώς και με τους κόμβους (node_id) που κάθε ένα από αυτά χρησιμοποιεί.

Ανάθεση και απόσυρση IP διευθύνσεων: Η διαδικασία της ανάθεσης και απόσυρσης των UOA στους διάφορους χρήστες των slices παρουσιάζεται στο παρακάτω σχήμα.



Ένας χρήστης ενός slice στέλνει αίτημα στον SliceMan, ο οποίος με τη σειρά του απευθύνεται στον επιλεγμένο κόμβο. Ο κόμβος αποδίδει ένα τοπικό user_id στο slice και μετά επιστρέφεται από τον SliceMan η διεύθυνση – uo_adrr και συνδέεται με τον χρήστη (user_id) καθώς και με ένα φυσικό δικτυακό interface του δικτύου με πρόθεμα (np).

Μεταφορά διεύθυνσης: Όταν ένα sliver μεταφέρεται από έναν φυσικό κόμβο σε έναν άλλο χρειάζεται να μεταφερθεί και η UOA διεύθυνση. Σε περίπτωση που το sliver μεταφέρεται σε διαφορετικό υποδίκτυο, δηλαδή αλλάζει το δικτυακό πρόθεμα (np), τότε σύμφωνα και με τον κανόνα παραγωγής των UOA διευθύνσεων η UOA διεύθυνση αλλάζει. Όταν το sliver μεταφέρεται εντός του ίδιου υποδικτύου, απλά μεταφέρεται χωρίς αλλαγή και η UOA διεύθυνση. Η διαδικασία που λαμβάνει χώρα παρουσιάζεται στο παρακάτω σχήμα



Αρχικά δημιουργείται ένα sliver στο κόμβο που θα γίνει η μεταφορά δίνοντας του ως όρισμα ένα user_id και στην συνέχεια γίνεται η αντιγραφή του περιεχομένου. Η uo_addr μπορεί να αποδοθεί στο user_id λίγο αργότερα αλλά δεν ενεργοποιείται σε ένα συγκεκριμένο interface πριν ο κόμβος προέλευσης διαγράψει τη διεύθυνση. Χρειάζεται ένας αυστηρός μηχανισμός συγχρονισμού όταν ο κόμβος προορισμού και ο κόμβος προέλευσης βρίσκονται στο ίδιο υποδίκτυο.

2.4 Κατηγορίες εικονικοποίησης

Μπορούμε να κατατάξουμε τις τεχνικές εικονικοποίησης ανάλογα με το υλικό στο οποίο εφαρμόζονται και ανάλογα με το εύρος εφαρμογής τους. Εκτός από την δικτυακή εικονικοποίηση που περιγράφηκε παραπάνω η εικονικοποίηση λαμβάνει χώρα και σε πολλούς άλλους τομείς του IT. Έτσι έχουμε την εικονικοποίηση εξυπηρετητή (server virtualization), την εικονικοποίηση μνήμης (memory virtualization) και την εικονικοποίηση χωρητικότητας (storage virtualization) όσο αφορά το υλικό στο οποίο εφαρμόζεται η εικονικοποίηση. Όταν αναφερόμαστε σε εικονικοποίηση εξυπηρετητή διακρίνουμε την πλήρη, μερική καθώς και την παραεικονικοποίηση συστήματος.

Server Virtualization: Στην συγκεκριμένη κατηγορία εικονικοποίησης το χρησιμοποιούμενο λογισμικό εξομοιώνει διαφορετικές τεχνολογικές εφαρμογές. Το μεγαλύτερο όφελος από τη λύση αυτή είναι ότι μια επιχείρηση μπορεί να αποκτήσει σημαντικά μεγαλύτερη υπολογιστική ισχύ, και είναι σε θέση να χρησιμοποιεί παλαιότερες εφαρμογές (legacy) και να προσομοιώνει κατανεμημένες εφαρμογές δικτύου πάνω στον ίδιο φυσικό διακομιστή, αφού ακόμα και διακομιστές παλαιότερης γενιάς μπορούν να φανούν χρήσιμοι. Προκειμένου να μειωθεί το κόστος απόκτησης συστημάτων υπάρχει η δυνατότητα εκτέλεσης εικονικών μηχανών εντός αυτών κάθε μια εκ των οποίων έχει το δικό της λειτουργικό σύστημα.

Η εικονικοποίηση εξυπηρετητή υλοποιείται με τέσσερις διαφορετικούς τρόπους ως:

- **πλήρης εικονικοποίηση (Full Virtualization):** Στη συγκεκριμένη περίπτωση γίνεται πλήρης εξομοίωση του υλικού του φυσικού συστήματος ώστε το πλήρως εικονικοποιημένο σύστημα να μπορεί να υποστηρίζει την εκτέλεση ενός λειτουργικού συστήματος χωρίς τροποποίηση στον πυρήνα του. Κατά την πλήρη εικονικοποίηση συστήματος εκτελούνται εικονικές μηχανές εντός των οποίων τρέχει και από ένα λειτουργικό σύστημα. Έτσι είναι σαν να έχουμε πολλούς υπολογιστές μέσα σε έναν. Η τεχνική της πλήρους εικονικοποίησης ενδείκνυται στις περιπτώσεις εκείνες που επιδιώκεται η χρήση λειτουργικών συστημάτων που δεν επιδέχονται αλλαγές στο πυρήνα τους όπως τα Windows.
- **μερική εικονικοποίηση (Partial Virtualization):** Σε αυτή την περίπτωση γίνεται μερικής εξομοίωση του συστήματος ώστε να μπορεί πάνω στο μερικώς

εικονικοποιημένο σύστημα να τρέξει ένα λειτουργικό με ορισμένες τροποποιήσεις. Δεν μπορεί να εκτελεστεί ολόκληρο το λειτουργικό σύστημα εντός μια εικονικής μηχανής αλλά μερικές εφαρμογές. Πολλά σύγχρονα λειτουργικά συστήματα όπως τα Windows και το Unix χρησιμοποιούν αυτήν την τεχνική εικονικοποίησης. Σημαντικό ρόλο στην μερική εικονικοποίηση διαδραματίζει η εικονικοποίηση του κενού χώρου διευθύνσεων.

- **παραεικονικοποίηση (Paravirtualization):** Το παραεικονικοποιήμενο περιβάλλον απαιτεί τροποποίηση στο πυρήνα του λειτουργικού σύστημα που εκτελείται σε αυτό και κατά συνέπεια η μέθοδος αυτή ενδείκνυται όταν επιθυμούμε να εγκαταστήσουμε λειτουργικό ανοικτού κώδικα του οποίου ο πυρήνας δύναται να υποστεί τροποποίηση
- **container-based εικονικοποίηση:** Η κατηγορία αυτή της εικονικοποίησης βασίζεται σε κατάτμηση του πυρήνα ενός λειτουργικού συστήματος σε οντότητες που ονομάζονται containers. Άλλη ονομασία της τεχνικής αυτής είναι operating system virtualization και αναφέρεται στην κατάτμηση του πυρήνα σε πολλαπλά user-spaces. Το λειτουργικό σύστημα διαμερίζεται σε οντότητες που ονομάζονται VEs, VPS ή jails και λειτουργούν ως πραγματικοί servers. Στο UNIX το operating system virtualization μπορεί να θεωρηθεί ως επέκταση του chroot μηχανισμού. Για την διασφάλιση απομόνωσης μεταξύ των διάφορων containers ο πυρήνας του κάθε λειτουργικού παρέχει διαχείριση των διαθέσιμων πόρων. Βασικά χαρακτηριστικά και δυνατότητες της εικονικοποίησης σε επίπεδο λειτουργικού συστήματος είναι τα ακόλουθα:

1. Μέσω live migration υπάρχει δυνατότητα ένα container να εκτελεστεί σε ένα άλλο κόμβο του ίδιου cluster με σκοπό την διασφάλιση μιας ισορροπίας στον φόρτο εργασίας κάθε κόμβου.
2. Πολλές φορές χρειάζεται να εκτελεστούν κάποιες εφαρμογές σε ξεχωριστά container η κάθε μια ώστε να εξασφαλιστεί ασφάλεια, αυτονομία από πλευράς hardware καθώς και δυνατότητα διαχείρισης των πόρων
3. Το συγκεκριμένο είδος virtualization εισάγει ελάχιστη ή και μηδαμινή επιβάρυνση (overhead), διότι τα προγράμματα που εκτελούνται στο virtual partition χρησιμοποιούν τις διεπαφές των κλήσεων συστήματος και δεν χρειάζεται να τρέξουν εντός κάποιας virtual machine όπως στην περίπτωση του full virtualization
4. Η χρήση της operating system virtualization θέτει περιορισμούς σε αντίθεση με άλλες τεχνικές virtualization όσο αφορά το λειτουργικό σύστημα που θα εκτελείται στα containers. Κάθε container δεν μπορεί να διαθέτει διαφορετικό λειτουργικό από αυτό στο οποίο υφίσταται τα containers. Ο περιορισμός αυτός αίρεται στην περίπτωση του

Solaris που επιτρέπει την εκτέλεση σε container ενός περιβάλλοντος που προσομοιώνει την 2.4 έκδοση του Linux ή μια παλιότερη έκδοση του Solaris.

Memory Virtualization: Η συγκεκριμένη περίπτωση virtualization αφορά [5] την ενοποίηση της μνήμης RAM κάθε μεμονωμένου συστήματος ενός datacenter σε μια εικονική που είναι προσπελάσιμη από κάθε υπολογιστικό σύστημα σε ένα δεδομένο υπολογιστικό σύμπλεγμα (cluster). Η κοινή μνήμη μπορεί να χρησιμοποιηθεί ως cache memory, ως μέσο μετάδοσης μηνυμάτων (messaging layer) η ως μνήμη για την CPU. Στη συγκεκριμένη περίπτωση εικονικοποίησης παρόλο που τα δεδομένα είναι αποθηκευμένα στις μνήμες RAM και τους σκληρούς δίσκους περισσότερων του ενός συστημάτων για την εικονική μνήμη είναι αποθηκευμένα συνεχόμενα και σε σειρά.

Τα πλεονεκτήματα που προκύπτουν από την εικονικοποίηση μνήμης είναι τα ακόλουθα:

- Αυξάνεται η χρησιμοποίηση μνήμης λόγω του διαμοιρασμού της σε πολλά υπολογιστικά συστήματα
- Μπορεί να μεταβάλλεται με την προσθήκη ή αφαίρεση πόρων στο υπολογιστικό σύμπλεγμα (cluster). Επιτρέπει σε εφαρμογές ή εξυπηρετητές που χρειάζονται κοινά δεδομένα να μπορούν να τα χρησιμοποιήσουν χωρίς να χρειάζονται αντίγραφα των δεδομένων αυτών ελαχιστοποιώντας έτσι τις συνολικές ανάγκες σε μνήμη.
- Συναντάται μικρότερη καθυστέρηση από άλλες λύσεις όπως SSD, SAN και NAS.

Storage Virtualization: Σε αυτή την περίπτωση εικονικοποιούμε τον χώρο αποθήκευσης. Οι διάφοροι χώροι αποθήκευσης που υφίστανται σε ένα cluster εμφανίζονται ως μία συσκευή την οποία μπορούμε να διαχειριζόμαστε από μια κονσόλα. Δημιουργούμε έναν εικονικό χώρο που αποτελείται από blocks τα οποία μπορούν να προσπελαστούν με χρήση δύο αναγνωριστικών, του LUN (Logical Unit Identifier) και ενός offset (LBA – Logical Block Adress) για το δεδομένο LUN. Κατά τον ίδιο τρόπο τακτοποιούνται τα δεδομένα και στα φυσικά μέσα αποθήκευσης. Μέσω ενός λογισμικού εικονικοποίησης έχει κανείς πρόσβαση σε ολόκληρο τον χώρο αποθήκευσης εικονικό και φυσικό. Το συγκεκριμένο λογισμικό έχει έναν πίνακα με καταχωρήσεις – αντιστοιχήσεις μεταξύ των LUN και offset στην εικονική και στην φυσική μνήμη που ονομάζονται meta-data . Έτσι όταν δημιουργηθεί ένα αίτημα για ανάσυρση δεδομένων από τον εικονικό δίσκο με δοσμένο το LUN και το LBA γίνεται αίτημα προς τον φυσικό χώρο αποθήκευσης με τα αντίστοιχα LUN και LBA του φυσικού χώρου αποθήκευσης όπως αυτά προκύπτουν από τα meta-data.

Στα πλεονεκτήματα του storage virtualization συγκαταλέγονται το migration δεδομένων από ένα αποθηκευτικό μέσο σε ένα άλλο του εικονικοποιημένου cluster και η απλούστευση των διαδικασιών backup και recovery για τον διαχειριστή. Βασικό πλεονέκτημα της εικονικοποίησης του χώρου αποθήκευσης είναι ότι υπάρχει ελευθερία αποθήκευσης των δεδομένων στο μέσο επιλογής χωρίς αυτό να γίνεται αντιληπτό από τον χρήστη. Επίσης επιτρέπει την αποθήκευση δεδομένων σε διάφορα φυσικά μέσα αποθήκευσης που το καθένα βασίζεται σε διαφορετικές τεχνολογίες. Με την εικονικοποίηση επιτυγχάνεται ουσιαστικά ενοποίηση του χώρου των δεδομένων κάθε φυσικού μέσου αποθήκευσης. Όταν προστίθεται ένα φυσικό μέσο αποθήκευσης συγκεκριμένης χωρητικότητας αυξάνεται και ο εικονικός χώρος αποθήκευσης κατά την ίδια χωρητικότητα. Επιπλέον επιτυγχάνεται η μεταφορά δεδομένων μεταξύ μέσων αποθήκευσης εντός του ίδιου περιβάλλοντος αποθήκευσης εξασφαλίζοντας παράλληλα αδιάλειπτη λειτουργία δεδομένου ότι τα δεδομένα διατηρούν το LUN και το LBA που είχαν αντικαθίσταται με τον καινούργιο που αποκτούν στον νέο χώρο αποθήκευσης.

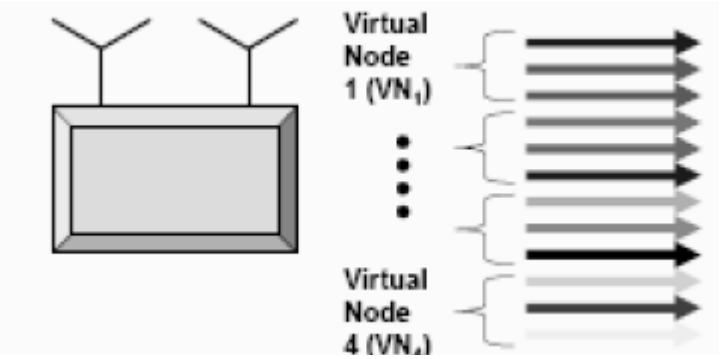
2.5 Wireless virtualization

Εκτός από τις τεχνικές ενσύρματης εικονικοποίησης έχουν αναπτυχθεί και τεχνικές εικονικοποίησης για ασύρματα συστήματα.

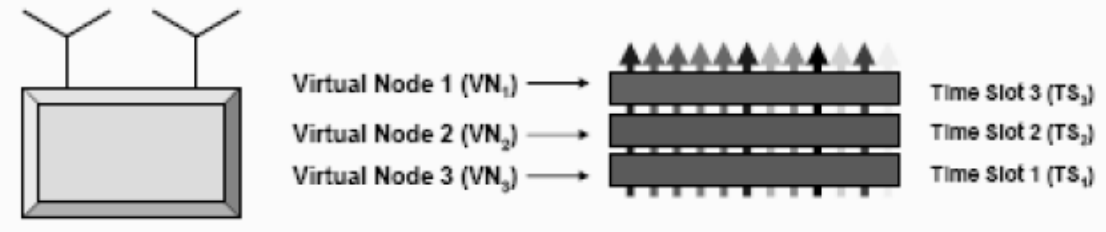
Το wireless virtualization συνίσταται στην δημιουργία εικονικών μηχανημάτων σε ασύρματους κόμβους δικτύων που υποστηρίζουν την οικογένεια πρωτοκόλλων 802.11 καθώς και στην δημιουργία εικονικών ζεύξεων μεταξύ των διάφορων εικονικών μηχανημάτων. Ουσιαστικά το virtualization ενός ασύρματου δικτύου οδηγεί στον διαμοιρασμό του bandwidth κάθε φυσικής ασύρματης ζεύξης σε λογικές ζεύξεις που διασυνδέουν τα διάφορα εικονικά μηχανήματα των φυσικών κόμβων. Βασικός στόχοι του wireless virtualization είναι η μεγιστοποίηση της χρήσης των υφιστάμενων πόρων χωρίς πτώση της ποιότητας υπηρεσίας, που συνίσταται κυρίως στην καθυστέρηση (delay) των πακέτων και στην διακύμανση καθυστέρησης (jitter), αλλά και η δημιουργία πειραματικών πλατφόρμων (testbeds) για αποτίμηση ασύρματων αρχιτεκτονικών και τεχνολογιών.

2.5.1 Τεχνικές εικονικοποίησης σε ασύρματα δίκτυα

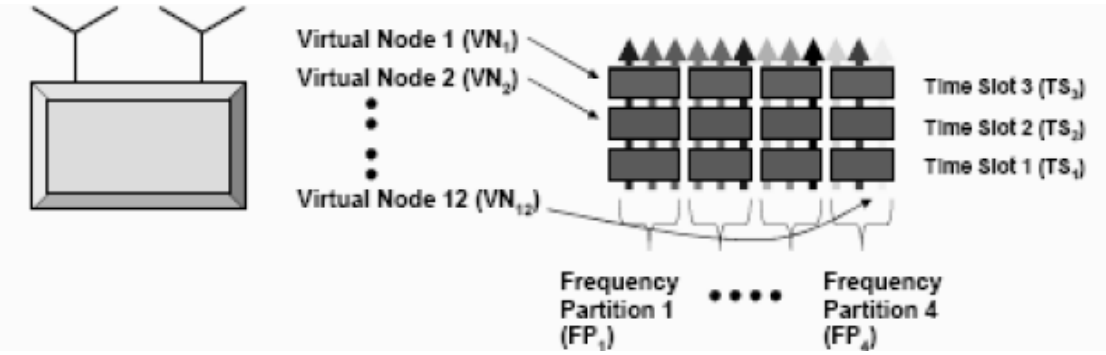
1)FDMA (Frequency Division Multiple Access): Δημιουργούμε εικονικές μηχανές στον κόμβο κατανέμοντας το φάσμα συχνοτήτων. Για παράδειγμα αν έχουμε διαθέσιμες 12 ορθογωνικές συχνότητες και θέλουμε να δημιουργήσουμε 4 εικονικά αντίγραφα του κόμβου τότε αποδίδουμε από 3 συχνότητες στην κάθε virtual machine. Όμως κάθε κάρτα απαιτεί κάποιο χρόνο για την αλλαγή καναλιού (πχ 5ms για τις κάρτες Atheros). Κάθε εικονικός κόμβος παίρνει την σειρά του με κυκλική εναλλαγή με βάση τον αλγόριθμο Round Robin. Υπάρχει δυνατότητα παραμετροποίησης του χρόνου για τον οποίο κάθε εικονικός κόμβος – μηχανή είναι ενεργός. Αν ονομάσουμε τον συγκεκριμένο χρόνο x τότε για τις κάρτες Atheros ο χρόνος που απαιτείται για να αποκτήσει ξανά ένα δεδομένο virtual machine δικαίωμα εκπομπής είναι (5+x)*n ms (cycle time). Αν χρησιμοποιηθούν πολλαπλές κάρτες ελαχιστοποιείται το cycle time όμως εμφανίζεται παρεμβολή κοινού καναλιού (co-channel interference).



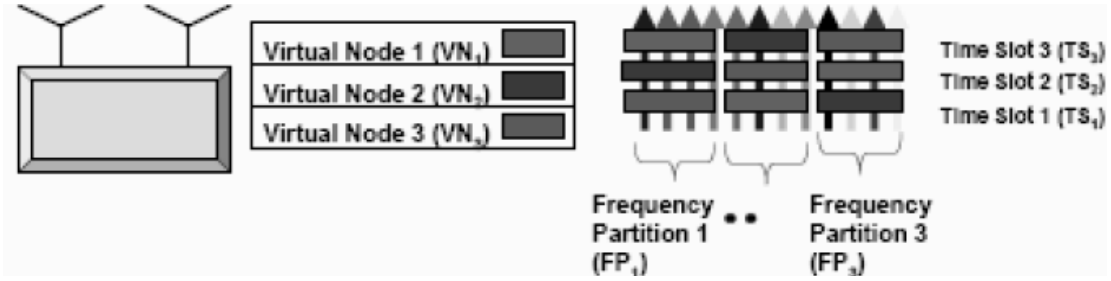
2)TDMA (Time Division Multiple Access): Στην τεχνική αυτή, οι χρήστες – virtual machine μοιράζονται μια συγκεκριμένη συχνότητα και ο καθένας καταλαμβάνει από μια χρονοθυρίδα (time slot). Στην περίπτωση του 802.11a για να δημιουργήσουμε 3 virtual machine και να κάνουμε χρήση της TDMA τεχνικής χρειάζεται μια συχνότητα και η κατανομή κάθε VM σε συγκεκριμένο time slot. Το cycle time είναι (10+x)*n ms όπου έχει συνυπολογιστεί ένας χρόνος από 1-10ms που χρειάζεται το Linux για την μετάβαση(switching) από το ένα virtual machine στο άλλο. Σημαντικό μειονέκτημα αυτής της τεχνικής είναι ότι όσο αυξάνεται το πλήθος των VMs ανά φυσικό κόμβο αυξάνονται οι απαιτούμενες χρονοθυρίδες και κατά συνέπεια ο χρόνος εκπομπής του πακέτου.



3)Συνδυασμός FDMA και TDMA: Οι χρήστες μπορούν να κατανέμονται σε ένα φάσμα συχνοτήτων και να χρησιμοποιούν μια συγκεκριμένη χρονοθυρίδα. Για παράδειγμα σε έναν 802.11a κόμβο με 12 ορθογωνικές συχνότητες μπορούν να κατανεμηθούν σε 4 λογικά group των 3 συχνοτήτων το καθένα. Τα λογικά group μπορούν και αυτά με την σειρά τους να κατατμηθούν σε χρονοθυρίδες. Έτσι κάθε χρήστης που χρησιμοποιεί από ένα VM κάνει χρήση συγκεκριμένου τμήματος του φάσματος και συγκεκριμένης χρονοθυρίδας. Για τον υπολογισμό του cycle time σε αυτή την τεχνική θα πρέπει να λάβουμε υπόψη το χρόνο που χρειάζεται μια κάρτα για να αλλάξει κανάλι και τον χρόνο που χρειάζεται το λειτουργικό για την επιλογή δεδομένης χρονοθυρίδες. Για χρόνους 5ms στις Atheros cards και 1-10ms στο Linux το cycle time υπολογίζεται $(5+(x+10)\cdot m)\cdot n$, όπου m είναι το πλήθος των χρηστών σε κάθε ένα από τα λογικά group όσες δηλαδή και οι απαιτούμενες χρονοθυρίδες και n το πλήθος των λογικών group.

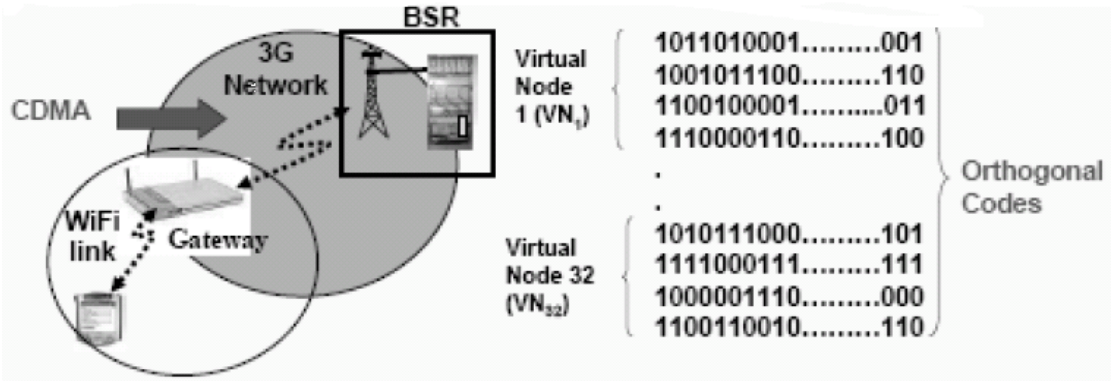


4)Μεταπήδηση συχνότητας (Frequency hopping): Στην τεχνική αυτή αντίθετα με την προαναφερθείσα οι χρήστες μπορούν να αλλάζουν συχνότητα και χρονοθυρίδα. Για κάθε virtual node δεσμεύονται ζεύγη συχνότητας και χρονοθυρίδας. Θεωρώντας πάλι έναν κόμβο 802.11a με 4 λογικά group συχνοτήτων έστω ένας virtual node VN1 που σε 4 διαδοχικές μεταδόσεις ακολουθεί τους ακόλουθους συνδυασμούς συχνότητας και χρονοθυρίδας (FP1,TS3), (FP2,TS3), (FP3,TS2) και (FP4,TS1) όπως παρουσιάζονται στο ακόλουθο σχήμα:



5)Πολλαπλή πρόσβαση με χρήση διαίρεσης κώδικα (CDMA)

Στην τεχνική αυτή γίνεται εικονοποίηση ενός κόμβου τμηματοποιώντας κατά τη «διάσταση» του κώδικα. Βρίσκει εφαρμογή σε σταθμού βάσης που λειτουργούν χρησιμοποιώντας ορθογώνιους κώδικες.



Για παράδειγμα, ένας CDMA2000 1x κόμβος υποστηρίζει το μέγιστο 128 ορθογώνιους κώδικες. Ως εκ τούτου μπορεί να τμηματοποιηθεί σε N εικονικούς κόμβους, αναθέτοντας M ορθογώνιους κώδικες για καθένα εικονικό κόμβο, όπου NxM=128, όπως φαίνεται στο επόμενο σχήμα.. στο σχήμα είναι M=4 και N=32.

Σε ένα υβριδικό 3G-WiFi δίκτυο ένα slice μπορεί να αποτελείται από ένα σύνολο ορθογώνιων κωδίκων μεταξύ του δρομολογητή στο 3G σταθμό βάσης και της πύλης και από μια τμηματοποίηση συχνότητας μεταξύ της πύλης και της συσκευής του τελικού χρήστη. Το CDMA αναφέρεται στην μεταγωγή κωδίκων μέσα σε ένα φυσικό κόμβο ώστε να προσομοιωθούν πολλαπλοί εικονικοί κόμβοι.

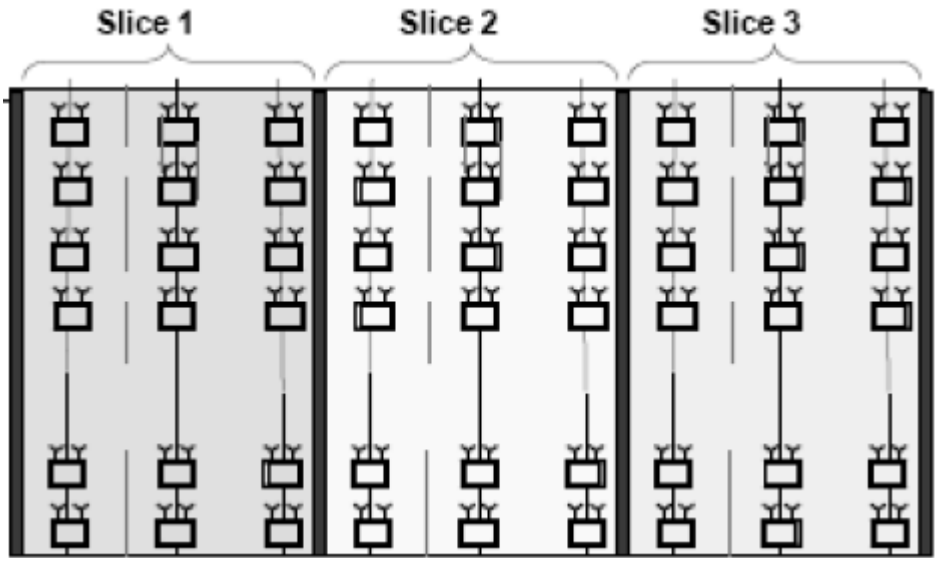
2.5.2 Μέθοδοι δημιουργίας slices

Λόγω ιδιαιτεροτήτων της ασύρματης τεχνολογίας η δημιουργία slices είναι διαφορετική σε ένα ασύρματο δίκτυο από ότι σε ένα ενσύρματο. Οι βασικότεροι περιορισμοί είναι οι εξής:

- Πολλά πειράματα σε ασύρματα τεχνολογίες αφορούν το φυσικό στρώμα και το στρώμα ζεύξης δεδομένων όμως ο διαμοιρασμός ενός κοινού κόμβου στα ασύρματα δίκτυα προς αξιοποίηση σε πολλά πειράματα καθίσταται αδύνατος.
- Τα πειράματα που αφορούν την αποτίμηση της κινητικότητας και της διαπομπής απαιτούν πολύ μικρή καθυστέρηση (latency) που είναι δύσκολο να επιτευχθεί όταν υπάρχει διαμοιρασμός του μέσου προς αξιοποίηση του και σε άλλα πειράματα
- Η έννοια της τοπολογία όπως αναπτύσσεται στα ασύρματα δίκτυα δεν είναι ταυτόσημη με αυτή των ενσύρματων δικτύων. Για την ακρίβεια στα ασύρματα δίκτυα δεν είναι όλοι οι κόμβοι ομότιμοι και τα πειράματα σε ασύρματες υποδομές εξαρτώνται σε μεγάλο βαθμό από την τοπολογία λόγω των φαινομένων διάδοσης. Για παράδειγμα διαφορετικά αποτελέσματα θα έχει ένα πείραμα με 1 σημείο πρόσβασης (access point) και 2 laptop πάνω σε voip τεχνολογίες όταν οι κόμβοι είναι κόντα και διαφορετικά όταν είναι μακριά από το access point.

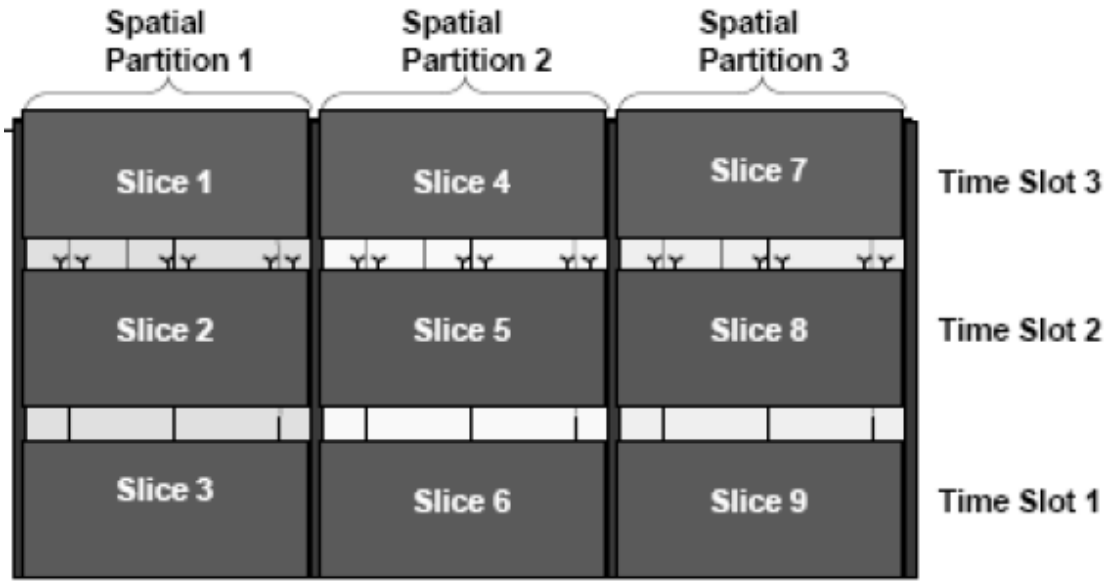
Στην συνέχεια παρουσιάζονται τρόποι δημιουργίας slices που βασίζονται στις τεχνικές virtualization ασύρματων δικτύων που περιγράφηκαν παραπάνω

1)Πολλαπλή πρόσβαση διαίρεσης χώρου (Space Division Multiple Access-SDMA): Στην τεχνική SDMA κάθε χρήστης χρησιμοποιεί πλήρως έναν κόμβο που σημαίνει ότι δεν εφαρμόζεται virtualization. Για παράδειγμα έστω ένα υπολογιστικό πλέγμα με κόμβους που επικοινωνούν με την οικογένεια πρωτοκόλλων 802.11. Οι ασύρματοι κόμβοι θα καταμεριστούν έτσι ώστε μόνο κόμβοι που βρίσκονται εντός ενός slice να μπορούν να επικοινωνούν. Κόμβοι που ανήκουν σε διαφορετικά slices δεν επικοινωνούν. Έτσι αν χωρίσουμε ένα grid σε 3 slices μας επιτρέπεται να εκτελέσουμε συνολικά 3 πειράματα από ένα και μόνο (λόγω έλλειψης virtualization) σε κάθε slice.

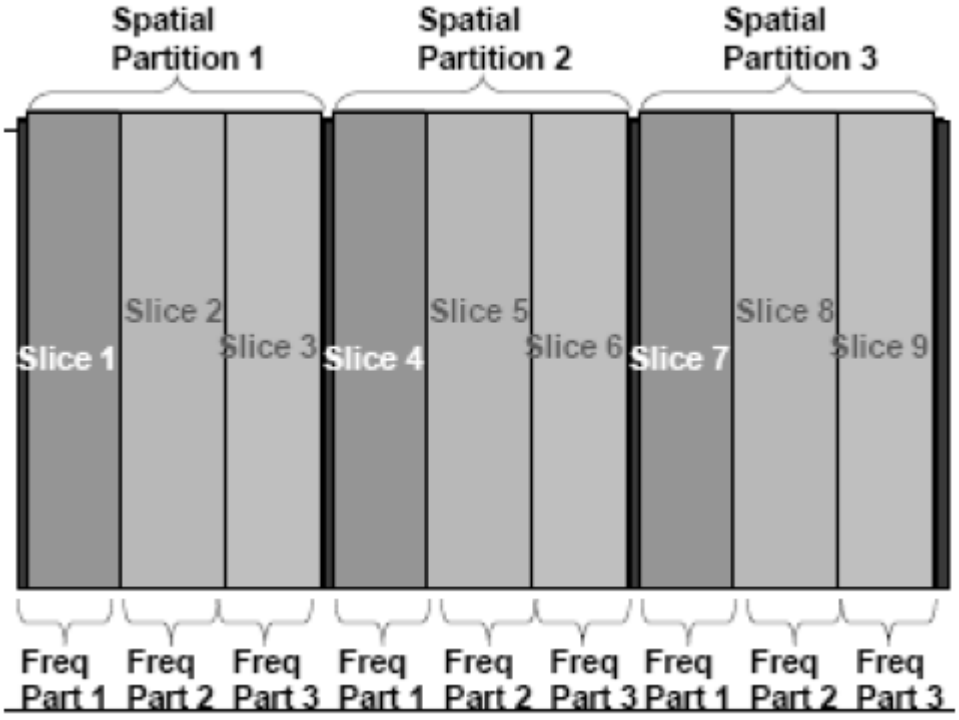


Σε ορισμένα testbeds όπως το ORBIT όπου οι φυσικοί κόμβοι είναι πολύ κοντά ο ένας με τον άλλο δεν είναι εύκολο ο διαχωρισμός σε slices μέσω φυσικού διαχωρισμού. Σε τέτοιες περιπτώσεις εφαρμόζεται η τεχνική του «artificial stretching’ που συνίσταται στον έλεγχο ισχύος και στη χρήση noise injection ώστε να τεθούν artificial εμπόδια μεταξύ κόμβων που ανήκουν σε διαφορετικά slices. Στην τεχνική SDMA λόγω έλλειψης virtualization ένας κόμβος ανήκει εξολοκλήρου σε ένα slice.

2)Συνδυασμένο SDMA με TDMA: Σε αυτή την περίπτωση οι φυσικοί κόμβοι διαχωρίζονται σε slices μέσως της προαναφερθείσας τεχνικής SDMA και σε κάθε slice κάθε κόμβος επικοινωνεί με τον άλλο μέσω χρονοθυρίδων. Για παράδειγμα για 3 partition και χρήση 3 time slot σε κάθε ένα έχουμε 9 πειράματα που μπορούν να εκτελεστούν ταυτόχρονα σε ένα grid.

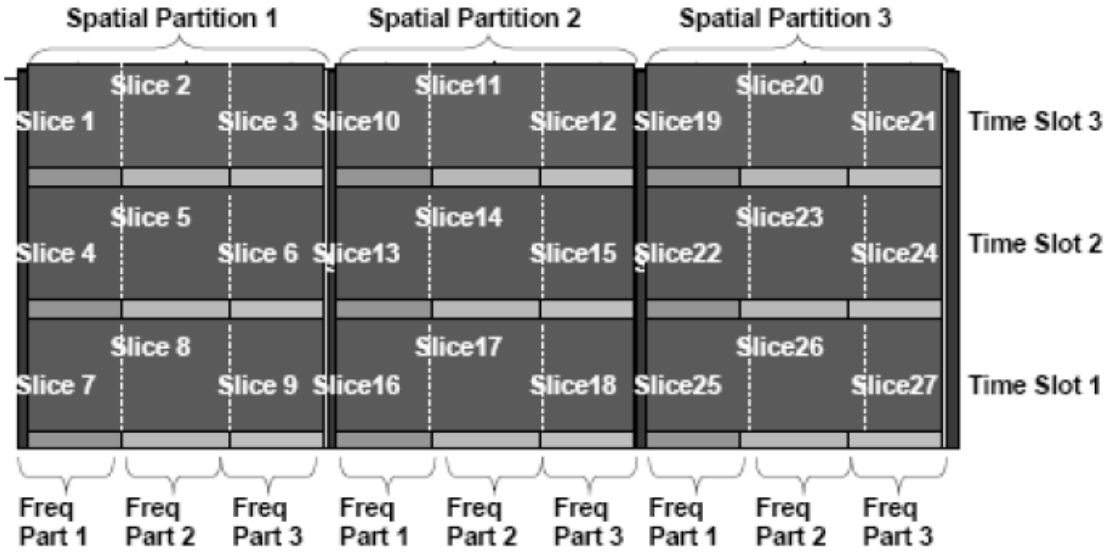


3)Συνδυασμένο SDMA και FDMA: Σε αυτή την τεχνική ένα πλέγμα από ασύρματους κόμβους διαχωρίζεται με spatial διαχωρισμό και κάθε κόμβος σε ένα slice επικοινωνεί με τους υπόλοιπους εντός του ίδιου slice με χρήση FDM διαμόρφωσης. Για παράδειγμα για 3 slices που κάθε ένα χρησιμοποιεί 3 συχνότητες επιτυγχάνουμε την ταυτόχρονη εκτέλεση 9 πειραμάτων



4) Συνδυασμένο SDMA, FDMA και TDMA: Σε αυτή την τεχνική ένα grid από ασύρματους κόμβους διαχωρίζεται σε slices και οι κόμβοι εντός του ίδιου slice επικοινωνούν μεταξύ τους με συνδυασμό FDM και TDM διαμόρφωσης. Για παράδειγμα

θεωρώντας 3 slices που κάθε ένα χρησιμοποιεί 3 συχνότητες και κάθε συχνότητα 3 χρονοθυρίδες εξασφαλίζουμε με τον τρόπο αυτό την ταυτόχρονη εκτέλεση 27 πειραμάτων στο υπολογιστικό πλέγμα.



Εφαρμογή του Greedy Algorithm στο Wireless Virtualization:

Θεωρώντας SDMA με FDMA ως μέθοδο για την δημιουργία slices και την εκτέλεση πειραμάτων, ορίζουμε για κάθε κόμβο του υπολογιστικού πλέγματος έναν μονοδιάστατο πίνακα F_i n θέσεων όσα και τα frequency partitions. Για κάθε διαθέσιμο frequency partition η αντίστοιχη τιμή στον πίνακα είναι 1 ενώ είναι 0 για τα μη διαθέσιμα. Δεδομένου ότι τα μη διαθέσιμα χρησιμοποιούνται για εκτέλεση πειραμάτων όσα κελιά φέρουν την τιμή 0 αποτελούν μέρος της εικονικής τοπολογίας. Όταν το πείραμα για το οποίο έχει δεσμευθεί μια δεδομένη συχνότητα ολοκληρώνεται τότε το αντίστοιχο F_i παίρνει την τιμή 1. Κάθε φορά που ζητείται μια τοπολογία γίνεται άμεσα διαθέσιμη εφόσον οι εικονικοί κόμβοι είναι διαθέσιμοι.

Η εφαρμογή του παραπάνω αλγορίθμου οδηγεί σε μειωμένη χρησιμοποίηση των πόρων. Προκειμένου να βελτιωθεί η χρήση των πόρων προτείνεται ανά τακτά περιοδικά διαστήματα να γίνεται διακοπή των πειραμάτων και επαναλειτουργία του δικτύου ώστε να ξαναγίνουν όλα τα F_i διαθέσιμα.

Optimal algorithm:

Καλύτερη λύση φαίνεται είναι να επαναξιολογείται η κατανομή των πόρων και να ανακατανέμονται όταν ένα νέο αίτημα φθάνει. Όμως, η λύση αυτή απαιτεί την αποθήκευση

πληροφοριών κατάστασης εισάγοντας δυσκολίες στην επεκτασιμότητα του δικτύου. Ως ιδανικότερη λύση προκρίνεται ένα υβριδικός αλγόριθμος ανάμεσα στην Greedy και Optimal μέθοδο.

2.5.3 Κατηγορίες Wireless networks και κατάλληλες τεχνικές virtualization

1. Emulator Networks: Τα δίκτυα αυτά υλοποιούνται σε πλήρως ελεγχόμενα περιβάλλοντα έτσι ώστε τα πειράματα που εκτελούνται σε αυτά να δίνουν σταθερά συγκεκριμένα αποτελέσματα. Χαρακτηριστικά παραδείγματα αυτής της κατηγορίας είναι τα ORBIT, Whynet, Emulab κλπ.
2. Urban grid: Πρόκειται για δίκτυα που αποτελούνται από μεγάλο πλήθος ασύρματων κόμβων και συναντώνται σε αστικές περιοχές, η δε πρόσβαση σε αυτά μπορεί να γίνει εύκολα μέσω ενός laptop με κατάλληλη ασύρματη κάρτα.
3. Suburban hybrid: Τα συγκεκριμένα ασύρματα δίκτυα καλύπτουν ευρύτερες γεωγραφικές περιοχές. Δεν είναι όμως οικονομική η υλοποίηση τους λόγω του περιορισμένου φάσματος της 802.11 οικογένειας πρωτοκόλλων. Αντίθετα ποιο αποδοτική κρίνεται η υλοποίηση κυψελωτών δικτύων βασισμένων στο 802.11
4. Cognitive Radio networks
5. Sensor Networks: Τα περισσότερα από τα δίκτυα αισθητήρων που έχουν εικονικοποιηθεί έχουν πολύ συγκεκριμένη λειτουργικότητα. Υπάρχουν όμως και πιο ευρέως χρησιμοποιούμενα δίκτυα αισθητήρων όπως το Mica II mote που χρησιμοποιείται σε πολλαπλά πειράματα.

Ο ακόλουθος πίνακας περιλαμβάνει τις αντιστοιχίες μεταξύ των κατηγοριών ασυρμάτων δικτύων και των τεχνικών wireless virtualization.

	SDMA	SDMA+ TDMA	SDMA+ FDMA	SDMA+ FDMA+ TDMA	SDMA+FH	SDMA + CDMA
Emulator	Ναι	Ναι	Ναι	Ναι	Ναι	Όχι
Urban grid	Ναι (όλα τα slices δεν είναι ίσα)	Ναι (απαιτεί αυστηρό time sync)	Ναι	Ναι (απαιτεί αυστηρό time sync)	Ναι (απαιτεί αυστηρό time sync)	Όχι
Suburban grid	Ναι	Ναι	Όχι (tech dependent)	Όχι (tech dependent)	Όχι (tech dependent)	Ναι
Cognitive Radio	Ναι	Όχι	Όχι	Όχι	Όχι	Όχι
Sensors	Ναι	Θεωρητικά Ναι, Πρακτικά Όχι (περιορισμένη CPU/μνήμη)	Θεωρητικά Ναι, Πρακτικά Όχι (περιορισμένη CPU/μνήμη)	Θεωρητικά Ναι, Πρακτικά Όχι (περιορισμένη CPU/μνήμη)	Θεωρητικά Ναι, Πρακτικά Όχι (περιορισμένη CPU/μνήμη)	Όχι

Τα emulator networks και τα urban grids μπορούν να εικονικοποιηθούν με τον ίδιο τρόπο. Όμως το γεγονός ότι τα urban grids δέχονται και μη ελεγχόμενη κίνηση σε αντίθεση με τα emulator networks έχει ως αποτέλεσμα τα urban grids εμφανίζουν μικρότερη ποιότητα υπηρεσία σε σχέση με τα emulator networks. Το πλήθος των ασύρματων κόμβων σε ένα κάθε slice σε ένα emulator δίκτυο είναι το ίδιο σε αντίθεση με ένα urban grid όπου οι ασύρματοι κόμβοι κατανέμονται σε slices με βάση διάφορα κριτήρια όπως η πυκνότητα των μη ελεγχόμενων χρηστών με laptop/PDA που έχουν πρόσβαση σε κόμβους του slice και κατά συνέπεια επηρεάζουν την κίνηση του δικτύου.

Στην περίπτωση των suburban hybrid networks χρησιμοποιείται SDMA για την δημιουργία slices. Κάθε slice περιλαμβάνει ένα ή περισσότερους σταθμούς βάσης και ένα πλήθος κόμβων. Η χρήση TDMA πρόσβασης με εφαρμογή χρονοθυρίδων ενδείκνυται σε τέτοιου είδους δίκτυο. Όμως η χρήση FDMA και FH τεχνικών πρόσβασης δεν είναι κατάλληλες λόγω του ήδη υπάρχοντος καταμερισμού και επαναχρησιμοποίησης του φάσματος που υφίσταται στις κυψελωτές επικοινωνίες.

2.5.4 Αντιστοίχηση τεχνικών wireless virtualization με εφαρμογές

Κάθε virtualization τεχνική μπορεί να εφαρμοστεί σε πειράματα που έχουν συγκεκριμένα χαρακτηριστικά και απαιτήσεις. Τα πειράματα μπορούν γενικά να χωριστούν σε τρεις κατηγορίες:

- Μικρής διάρκειας μικρότερης από μια μέρα
- Μεσαίας διάρκειας μικρότερη από μήνα
- Μεγάλης διάρκειας μεγαλύτερης από μήνα

Μερικές από τις απαιτήσεις που έχουν συγκεκριμένα πειράματα αφορούν την καθυστέρηση (delay) καθώς και την διακύμανση της (jitter) ιδιαίτερα σε τεχνολογίες Voip που ενδιαφέρει τα πακέτα να καθυστερούν κατά τον ίδιο χρόνο. Σημαντική παράμετρος είναι και ο ρυθμός απωλειών πακέτων (packet loss rate) καθώς και το throughput σε εφαρμογές file sharing. Μια άλλη σημαντική παράμετρος είναι η κινητικότητα (mobility) είτε φυσική είτε προγραμματίσιμη

Εξαιτίας της απαίτησης σε πολλα πειράματα για πολύ μικρό delay και jitter από 1-10ms υπάρχει περιορισμός στη χρήση TDMA τεχνικής. Πειράματα που αφορούν το MAC στρώμα μπορούν να υλοποιηθούν με SDMA και SDMA+FDMA τεχνικές δημιουργίας slices. Τέτοια πειράματα μπορούν να είναι ο έλεγχος των πόρων, ο έλεγχος πρόσβασης, ποιότητας υπηρεσίας καθώς και ρύθμιση παραμέτρων όπως απενεργοποίηση των Ack. Τα πειράματα που αφορούν έλεγχο συχνότητας δεν μπορούν να υλοποιηθούν σε SDMA+FDMA slices. Σε ότι αφορά τα cross-layer πειράματα το SDMA θεωρείται η κατάλληλη τεχνική για δημιουργία slices. Η SDMA+FDMA τεχνική είναι κατάλληλη για πειράματα που δεν έχουν υψηλές απαιτήσεις για χαμηλό latency/jitter.

	SDMA	SDMA+ TDMA	SDMA+FDMA	SDMA+FDMA+ TDMA	SDMA+FH	SDMA+ CDMA
MAC πειράματα	Ναι	Ναι (limited)	Ναι (limited)	Ναι (limited)	Ναι (limited)	Ναι (limited)
Πολλή μικρή καθυστέρηση	Ναι	Όχι	Όχι	Όχι	Όχι	Όχι
Μικρή έως μεσαία καθυστέρηση	Ναι	Ναι (για περιορισμένο αριθμό slices)	Ναι (για περιορι-σμένο αριθ-μό slices)	Ναι (για περιορισμένο αριθμό slices)	Ναι (για περιορισμένο αριθμό slices)	Ναι
Χαμηλή απώλεια	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
Υψηλή διαπερατότητα	Ναι	Ναι (εξαρτάται από τον αριθμό πειραμάτων που μοιράζονται πόρους / επιθυμητή διαπερατότητα)	Ναι (εξαρτάται από τον αριθμό πειραμάτων που μοιράζονται πόρους / επιθυμητή διαπερατότητα)	Ναι (εξαρτάται από τον αριθμό πειραμάτων που μοιράζονται πόρους / επιθυμητή διαπερατότητα)	Ναι (εξαρτάται από τον αριθμό πειραμάτων που μοιράζονται πόρους / επιθυμητή διαπερατότητα)	Ναι

Μια άλλη σημαντική παράμετρος των πειραμάτων όπως προαναφέρθηκε είναι η κινητικότητα που αναφέρεται σε φυσική και σε προγραμματίσιμη.

Πειράματα που σχετίζονται με προγραμματίσιμη κινητικότητα μπορούν να υλοποιηθούν σε slices βασισμένα στην SDMA τεχνική ενώ εκείνα που σχετίζονται με φυσική κινητικότητα μπορούν να υλοποιηθούν με SDMA, TDMA, FDMA και FH.

	SDMA	SDMA+ TDMA	SDMA+ FDMA	SDMA+FDMA+TDMA	SDMA+FH	SDMA+CDMA
Mobility (programmed)	Ναι	Ναι (το όχημα έχει το δικό του χώρο)	Όχι	Όχι	Όχι	Ναι
Mobility (natural)	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
Μεταγωγή Καναλιού	Ναι	Ναι	περιορισμένη	περιορισμένη	περιορισμένη	περιορισμένη
Πολλαπλή διανομή	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι

2.5.5 Περιορισμοί των τεχνικών wireless virtualization

Κατά την παραπάνω μελέτη των τεχνικών wireless virtualization προέκυψαν περιορισμοί κατά την εφαρμογή κάθε τεχνικής. Παρακάτω συνοψίζονται οι βασικοί περιορισμοί σε κάθε περίπτωση.

FDMA:

1. Πλήθος των ορθογωνικών συχνοτήτων (3 για το 802.11b, 12 για τα 802.11a/g)
2. Χρόνος μετάβασης από την μία συχνότητα στην άλλη
3. Ενεργός χρόνος για κάθε εικονικό κόμβο (πλήθος μεταδιδόμενων πακέτων/χρόνο)
4. Συνδυασμός πειραμάτων με διαφορετικές απαιτήσεις σε throughput, latency
5. Πολλαπλές προσεγγίσεις βασισμένες σε κάρτες που υπόκεινται σε παρεμβολές κοινού καναλιού

SDMA:

1. Πλήθος των partition που δεν παρεμβάλουν το ένα το άλλο
2. Πλήθος των κόμβων σε κάθε partition

TDMA Overlay:

1. Χρόνος μετάβασης μεταξύ των συχνοτήτων
2. Χρόνος μετάβασης μεταξύ των χρονοθυρίδων σε μια συγκεκριμένη συχνότητα
3. Χρόνος χρήσης καναλιού (υποθέτοντας CSMA/CA)
4. Ενεργός χρόνος για κάθε εικονικό κόμβο (πλήθος μεταδιδόμενων πακέτων/χρόνο)
5. Συνδυασμός πειραμάτων με διαφορετικές απαιτήσεις σε throughput, latency

Frequency Hopping:

1. Πλήθος των ορθογωνικών συχνοτήτων (3 για το 802.11b, 12 για τα 802.11a/g)
2. Χρόνος μετάβασης από την μία συχνότητα στην άλλη
3. Χρόνος μετάβασης μεταξύ των χρονοθυρίδων σε μια συγκεκριμένη συχνότητα
4. Χρόνος χρήσης καναλιού
5. Ενεργός χρόνος για κάθε εικονικό κόμβο (πλήθος μεταδιδόμενων πακέτων/χρόνο)
6. Συνδυασμός πειραμάτων με διαφορετικές απαιτήσεις σε throughput, latency

Artificial Stretching:

1. Πλήθος των partitions που δεν παρεμβάλουν το ένα το άλλο
2. Πλήθος των κόμβων σε κάθε partition
3. τμηματοποίηση του ελέγχου με βάση την πηγή θορύβου
4. τμηματοποίηση του ελέγχου με βάση την ενέργεια εκπομπής/λήψης των κόμβων

Ένα χαρακτηριστικό project που αναπτύχθηκε με χρήση της wireless virtualization είναι το ORBIT που αναπτύσσεται σε άλλο κεφάλαιο της παρούσας εργασίας.

3. Τεχνολογίες εικονικοποίησης

Στο τρίτο κεφάλαιο θα περιγραφούν οι δομικοί λίθοι της εικονικοποίησης, οι τεχνολογίες δηλαδή εκείνες που επιτρέπουν την δημιουργία εικονικών περιβάλλοντων μέσω της εικονικοποίησης των συστημάτων και της δικτυακής εικονικοποίησης. Ανάλογα με την κατηγορία εικονικοποίησης (πλήρη εικονικοποίηση, paravirtualization, ή container-based εικονικοποίηση) χρησιμοποιούμε και τις αντίστοιχες τεχνολογίες για την εικονικοποίηση των εξυπηρετητών. Οι πιο αντιπροσωπευτικές από αυτές τις τεχνολογίες είναι η πρόταση της VMware που αναφέρεται σε πλήρη εικονικοποίηση, το Xen που αναφέρεται σε paravirtualization και τα jails όπως ονομάζονται τα containers στο λειτουργικό FreeBSD. Επίσης γίνεται αναφορά στους εικονικούς μεταγωγείς με χαρακτηριστικά παραδείγματα το Open vSwitch και το OpenFlow

3.1 Η πρόταση της VMware

Η VMWARE [3] είναι ένας από τους μεγαλύτερους κατασκευαστές λογισμικού virtualization παγκοσμίως. Η πλατφόρμα που ανέπτυξε η VMware για virtualization περιλαμβάνει το Vsphere ένα λογισμικό για την εικονικοποίηση των πόρων ενός x86 υπολογιστή (CPU, RAM, σκληρό δίσκο, κάρτα δικτύου) με σκοπό την δημιουργία μιας πλήρως λειτουργικής virtual machine που μπορεί να τρέξει το δικό της λειτουργικό σύστημα και εφαρμογές σαν πραγματικός υπολογιστής. Η πρόταση της VMware για το virtualization βασίζει την λειτουργία της σε ένα στρώμα λογισμικού που εγκαθίσταται είτε απευθείας στο hardware του υπολογιστή είτε σε ένα λειτουργικό σύστημα πάνω στο οποίο θα τρέξουν τα virtual machines. Το στρώμα λογισμικού περιέχει έναν επόπτη για τα virtual machines που ονομάζεται "hypervisor" ο οποίος είναι υπεύθυνος για την κατανομή των πόρων στα virtual machines και εξασφαλίζει ότι μπορούν να εκτελούνται στον ίδιο υπολογιστή πολλαπλά λειτουργικά συστήματα το καθένα σε συγκεκριμένο virtual machine ανεξάρτητα το ένα από το άλλο. Με το VMWARE και συγκεκριμένα με την πλατφόρμα VMware vsphere Hypervisor ESXi μπορεί κάποιος να εικονικοποιήσει από ένα απλό μεμονωμένο υπολογιστή μέχρι ολόκληρο datacenter που μπορεί να αποτελείται από εκατοντάδες υπολογιστές και μέσα αποθήκευσης. Ο χρήστης δεν χρειάζεται να αναθέτει στους εξυπηρετητές ή στο δίκτυο συγκεκριμένο εύρος ζώνης. Αντίθετα, η ανάθεση πόρων γίνεται με δυναμικό τρόπο όταν χρειάζονται από τις εφαρμογές που τρέχουν στο

εικονικοποιημένο περιβάλλον. Επιτρέπει επίσης τη διασύνδεση του εικονικοποιημένου περιβάλλοντος που ορίζει με άλλα τέτοια περιβάλλοντα γεγονός που οδηγεί σε μεγαλύτερες οικονομίες κλίμακας.

Η πλατφόρμα του VMware Virtualization μπορεί να θεωρηθεί ένας πολλαπλασιαστής πόρων. Καθιστά διαθέσιμα τα στοιχεία ενός x86 server σε πολλούς εξηπηρετητές εφαρμογών, αντί σε μόνο έναν όπως συνήθως.

Η πρόταση της VMware για το virtualization περιλαμβάνει μια πληθώρα εργαλείων για την διεκπεραίωση συγκεκριμένων λειτουργιών. Συγκεκριμένα η λύση της Vmware περιλαμβάνει τα Vmware VMotion, VMware vStorage Vmotion, VMware DRS και VMware HA. Αντίστοιχα εργαλεία παρέχουν και άλλες εταιρίες που προσφέρουν λύσεις virtualization.

3.1.1 VMware VMotion

Το VMware VMotion επιτρέπει τη μετακίνηση των virtual machines που βρίσκονται σε λειτουργία από ένα φυσικό server σε έναν άλλο εξασφαλίζοντας αδιάλειπτη λειτουργία. Η δυνατότητα αυτή επιτρέπει την μεταφορά virtual machines από φορτωμένους servers σε λιγότερο φορτωμένους. Με αυτόν τον τρόπο γίνεται καλύτερη αξιοποίηση των πόρων του δικτύου. Η μετακίνηση ενός virtual machine από έναν server σε έναν άλλο δεν γίνεται αντιληπτή από τον χρήστη που λαμβάνει υπηρεσίες από το datacenter. Το VMware VMotion είναι διαθέσιμο σε όλες τις εκδόσεις του VMware vSphere. Στο παρακάτω σχήμα απεικονίζεται η λειτουργία του VMware ESXi.

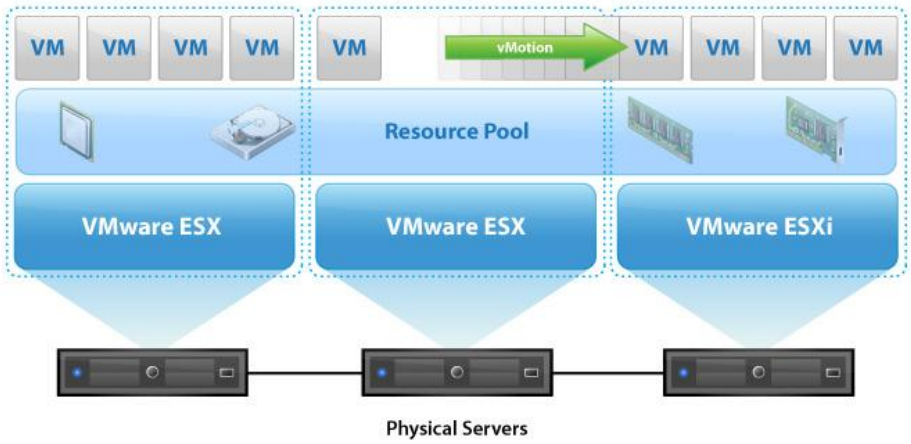
3.1.2 VMware Storage VMotion

Το Storage VMotion επιτρέπει την μετακίνηση των virtual machines disk files από ένα δίσκο σε έναν άλλο του virtualized περιβάλλοντος. Με χρήση του χαρακτηριστικού αυτού μπορούμε να ελαττώσουμε τον χρόνο κατά τον οποίο κάποια αρχεία δεν θα ήταν διαθέσιμα αν χρησιμοποιούταν κάποια παραδοσιακή μέθοδος μεταφοράς αρχείων. Το VMware Storage Vmotion λειτουργεί με την ίδια φιλοσοφία που διέπει και το VMware VMotion επιτρέποντας live migration των εικονικών μηχανημάτων.

3.1.3 VMware DRS

Το VMware DRS (Dynamic Resource Scheduler) επικεντρώνεται στον έλεγχο και τη διαχείριση των πόρων ενός virtual datacenter. Συγκεκριμένα, το DRS παρακολουθεί το φόρτο εργασίας (workload) των εν λειτουργία εικονικών μηχανημάτων και την χρησιμοποίηση πόρων των εξυπηρετητών έτσι ώστε τα εικονικά μηχανήματα να κατανέμονται σε εξυπηρετητές ανάλογα με τον φόρτο που φέρει το καθένα επιτυγχάνοντας έτσι υψηλή απόδοση και χρησιμοποίηση των πόρων του datacenter. Το DRS χρησιμοποιεί ένα δρομολογητή πόρων (resource scheduler) και σε συνδυασμό με το VMotion που αναλαμβάνει το live migration (μετακίνηση) των εικονικών μηχανημάτων από έναν server σε έναν άλλο οι φορτωμένοι servers μεταφέρουν ορισμένα εικονικά μηχανήματα σε λιγότερο φορτωμένους servers.

Όταν ένας νέος εξυπηρετητής προστεθεί στο datacenter τότε το DRS κάνει εκ νέου κατανομή των εικονικών μηχανημάτων ώστε να μοιράσει τον φόρτο εργασίας (workload) στους εξυπηρετητές του datacenter. Όταν ένας ήδη υπάρχον server πρέπει να τεθεί εκτός λειτουργίας το DRS κατανείμει το φορτίο του σε άλλους εξυπηρετητές. Όταν ένας server έχει διαθέσιμους τόσους πόρους ώστε να τρέξει όλα τα virtual machines ενός άλλου server τότε εκείνα μέσω live migration μεταφέρονται σε αυτόν και ο μη χρησιμοποιούμενος server τίθεται σε αναμονή. Με την τεχνική αυτή το DPM (Distributed Power Management) βελτιστοποιεί την κατανάλωση ισχύος στους servers του datacenter. Το DRM με κατάλληλη παραμετροποίηση μπορεί να εκτελεί αυτόματα ενέργειες που αφορούν τον ισομερισμό του φόρτου εργασίας μεταξύ των εξυπηρετητών. Παρακάτω απεικονίζεται η διαδικασία εξισορρόπησης του φορτίου μεταξύ των διάφορων εξυπηρετητών.



3.2 XEN

Το Xen Supervisor [4] είναι ένα λογισμικό που τρέχει απευθείας πάνω από το hardware αντικαθιστώντας στην ουσία το λειτουργικό σύστημα, επιτρέποντας την ταυτόχρονη εκτέλεση πάνω σε αυτό πολλών λειτουργικών συστημάτων (Linux, Netbsd, FreeBsd, Solaris, Windows)

Υποστηρίζει x86, x86-64, Itanium, Power PC και ARM επεξεργαστές.

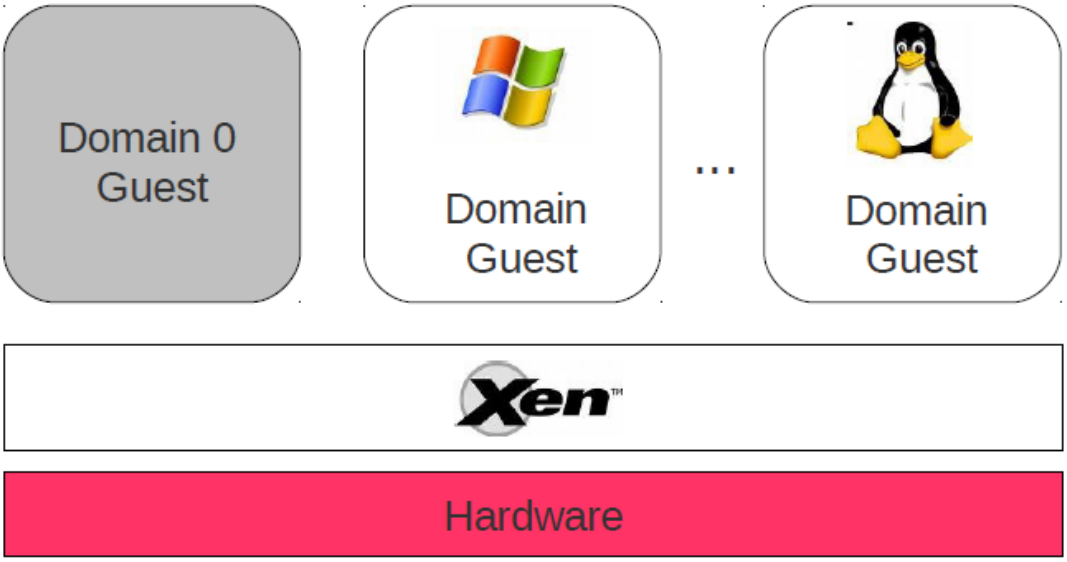
Το XEN χωρίζεται στα εξής επίπεδα:

Το domain0 είναι το επίπεδο εκείνο που έχει πρόσβαση στο hardware και έχουν πρόσβαση σε αυτό οι διαχειριστές του συστήματος για λειτουργίες όπως η εκκίνηση, ο τερματισμός είσοδοι/έξοδοι κτλ.

Τα domainU δρουν ανεξάρτητα το κάθε ένα και μπορούν να τρέχουν το καθένα από ένα λειτουργικό σύστημα πάνω στο domain0. Αν ο πυρήνας του λειτουργικού είναι τροποποιημένος ώστε να αναγνωρίζει ότι τρέχει πάνω στο hypervisor και όχι απευθείας πάνω στο λογισμικό τότε έχουμε paravirtualization, ενώ όταν το λειτουργικό «αγνοεί» ότι τρέχει πάνω στο hypervisor θεωρώντας ότι τρέχει απευθείας πάνω στο hardware τότε τα διάφορα domainU συνιστούν και από μια εικονική μηχανή που στη συγκεκριμένη

περίπτωση επειδή αναφερόμαστε σε λειτουργικό hardware virtual machine έχουμε απλή εικονικοποίηση. Όσοι χρησιμοποιούν τα domainU αναφέρονται ως μη διακεκριμένοι χρήστες (unprivileged guests) και δεν έχουν άμεση πρόσβαση στο υλικό, παρά μόνο ο διαχειριστής μέσω του domain0 όπως προαναφέρθηκε.

Το παρακάτω σχήμα αποδίδει την αρχιτεκτονική του Xen



Ο Xen Hypervisor τρέχει απευθείας πάνω στο hardware και είναι υπεύθυνος για τον χρονοπρογραμματισμό της CPU καθώς και τον καταμερισμό της μνήμης στις διάφορες εικονικές μηχανές που τρέχουν. Ο hypervisor αποτελεί το υλικό για τις εικονικές μηχανές. Μέσω του hypervisor δε είναι εφικτή κάθε πρόσβαση σε δίκτυο εξωτερική συσκευή, και οποιαδήποτε συσκευή εισόδου εξόδου. Οι λειτουργίες αυτές υλοποιούνται από το domain0.

Domain0: Το domain0 είναι η μόνη εικονική μηχανή που τρέχει πάνω από τον hypervisor με δυνατότητα πρόσβασης στο υλικό του συστήματος και επικοινωνίας με άλλες εικονικές μηχανές που τρέχουν πάνω στο hypervisor δηλαδή τα διάφορα domainU είτε PV ειτε HDV χρήστες μέσω δύο προγραμμάτων οδήγησης αντίστοιχα. Προκειμένου να ξεκινήσουν να τρέχουν τα διάφορα domainU απαιτείται η εκκίνηση το domain0 προηγουμένως.

Το domain0 περιλαμβάνει δύο προγράμματα οδήγησης για να δέχονται αιτήματα για το δίκτυο και τον σκληρό δίσκο από τους χρήστες. Ο NetWork Backend driver επικοινωνεί απευθείας με τον τοπικό δίκτυο για κάθε αιτούμενη προσπέλαση δικτύου από κάθε εικονική μηχανή ενώ ο block backend driver χρησιμοποιείται για αιτήματα που αφορούν την προσπέλαση του σκληρού δίσκου.

DomainU: Όπως προαναφέρθηκε τα domainU δεν έχουν καμία πρόσβαση στο υλικό του συστήματος. Οι παραεικονικοποιημένες μηχανές αυτές δηλαδή στις οποίες το λειτουργικό σύστημα λόγω κάποιων τροποποιήσεων αναγνωρίζει ότι τρέχει πάνω στον hypervisor καθώς και ότι τρέχουν και άλλες μηχανές παράλληλα αναφέρονται ως domainU PV guests. Αντίθετα οι πλήρως εικονικοποιημένες μηχανές που τρέχουν πάνω στον Hypervisor θεωρώντας ότι τρέχουν απευθείας πάνω στο υλικό που σημαίνει ότι αγνοούν την ύπαρξη άλλων εικονικών μηχανών που τρέχουν παράλληλα αναφέρονται ως domainU HVM guests. Ένας domainU PV guest περιλαμβάνει δύο προγράμματα οδήγησης ένα για πρόσβαση στο δίκτυο και ένα για πρόσβαση στο δίσκο που ονομάζονται PV Network Driver και PV Block Driver αντιστοίχως.

Αντιθέτως, ένας domainU HVM guest βασίζει την λειτουργία διαφορετικά. Συγκεκριμένα ένας daemon που ονομάζεται Qemu-dm τρέχει πάνω από το domain0 και όχι εντός της εικονικής μηχανής για κάθε HVM guest. Ο ρόλος του είναι να δέχεται αιτήματα για πρόσβαση στο δίκτυο και προσπέλαση του δίσκου.

Εντός του domain0 υπάρχει πληθώρα από daemons που χρησιμοποιούνται με στόχο στη διαχείριση και τον έλεγχο του εικονικοποιημένου περιβάλλοντος. Παρακάτω παρουσιάζεται μερικά σημαντικά daemons:

Xend: Είναι γραμμένο σε Python και θεωρείται ο διαχειριστής του συστήματος Xen. Με την βοήθεια της libxenctrl βιβλιοθήκης στέλνει αιτήματα στον Xen hypervisor μέσω xml rpc με χρήση του Xm εργαλείου

Xm: Είναι μια γραμμή εντολών που μεταφέρει τις εντολές του χρήστη στο Xend μέσω XML RPC

Xenstored: Στο Xenstored daemon αποθηκεύονται πληροφορίες που αφορούν συνδέσεις μεταξύ του domain0 και των domainU. Οι καταχωρημένες πληροφορίες χρησιμοποιούνται για επικοινωνία μεταξύ των διάφορων εικονικών μηχανών του συστήματος.

Libxenctrl: Η Libxenctril είναι μια βιβλιοθήκη σε γλώσσα C που επιτρέπει την επικοινωνία του Xend με το Zen Hypervisor μέσω του domain0. Ο privcmd driver μεταφέρει το αίτημα στον hypervisor.

Qemu-dm: Όπως αναφέρθηκε προηγουμένως κάθε HVM Guest που τρέχει στο Zen απαιτεί και από ένα Quemu daemon που είναι υπεύθυνο για τον χειρισμό των αιτημάτων για πρόσβαση στο δίκτυο ή στο δίσκο. Το Qemu πρέπει να είναι εκτός του Xen Hypervisor και

μάλιστα να τρέχει εντός του domain0 ώστε να μπορεί να χειρίζεται εντολές για προσπέλαση του δικτύου ή του δίσκου. Στην έκδοση 3.3 του Xen ένα νέο daemon το Stub-dm θα είναι διαθέσιμο για χρήση σε κάθε Domain U HVM Guest. Έτσι δεν θα χρειάζεται κάθε εικονική μηχανή να έχει το δικό της daemon.

Κάθε Domain U HVM Guest χρειάζεται από ένα Xen Virtual Firmware, ένα εικονικό BIOS, ώστε να μπορεί το λειτουργικό σύστημα κάθε guest να λαμβάνει τις απαραίτητες εντολές κατά την εκκίνηση του.
Όπως προαναφέρθηκε ο Xen Hypervisor δεν υποστηρίζει αιτήματα πρόσβασης στο δίκτυο ή στο δίσκο αλλά ένας παραεικονικοποιημένος domainU επικοινωνεί με την συσκευή μέσω του domain0 του Hypervisor. Για παράδειγμα στη περίπτωση που ένας παραεικονικοποιημένος domainU θέλει να γράψει δεδομένα στον τοπικό δίσκο, ο domainU block driver λαμβάνει το αίτημα και γράφει τα δεδομένα μεσω του Hypervisor στη τοπική μνήμη που μοιράζεται με το domain0. Στη συνέχεια μέσω ασύγχρονων διακοπών στον Xen Hypervisor το domain0 επικοινωνεί με το PV domainU. Το domain0 θα λάβει μια διακοπή από τον Xen Hypervisor με αποτέλεσμα ο PV Block Backend Driver του domain0 να προσπελάσει την τοπική μνήμη του συστήματος από την κοινή με την παραεικονικοποιημένο domainU μνήμη. Τα συγκεκριμένα δεδομένα από την κοινή μνήμη γράφονται αργότερα στον σκληρό δίσκο.
Ένα άλλο σημαντικό daemon tool είναι το Xen PCI Passthru που έχει σχεδιαστεί για να βελτιωθεί η επίδοση του Xen όσο αφορά τη πρόσβαση στο hardware. Πιο συγκεκριμένα επιτρέπει σε κάθε παραεικονικοποιημένο domainU να έχει άμεση πρόσβαση στο hardware χωρίς την μεσολάβηση του domain0.

3.3 USER-MODE LINUX (UML)

Το User-mode Linux [25]τρέχει ένα Linux virtual machine σε ένα σύνολο διεργασιών που βρίσκονται σε ενα host. Ένα UML virtual machine είναι ικανό να τρέχει σχεδόν το ίδιο σύνολο από διεργασίες συγκριτικά με το host. Η αρχιτεκτονική του παρέχει όλη την υποστήριξη σε χαμηλού επιπέδου υλικό την οποία χρειάζεται ο αρχικός πυρήνας και υλοποιείται αποκλειστικά από κλήσεις συστήματος. Είναι ικανό να τρέξει σχεδόν όλες τις εφαρμογές εκτός από:

- Διαδικασίες εγκατάστασης, οι οποίες ελέγχουν για υλικό χρησιμοποιώντας εξουσιοδοτημένες εντολές.
- Εξομοιωτές, όπως dosemu (εξομοιώνει το λειτουργικό σύστημα DOS), wine (επιτρέπει την εκτέλεση προγραμμάτων windows σε Linux, BSD, Solaris και MAC OS X) και plex86 (εξομοιώνει ένα x86 επεξεργαστή και επιτρέπει την εκτέλεση Linux λειτουργικού ως guest).

3.3.1 Συσκευές

Όλες οι συσκευές που είναι προσβάσιμες μέσα στο virtual machine είναι από μόνες τους εικονικές. Το UML υποστηρίζει ένα ευρύ φάσμα από συσκευές:

- **Κονσόλες και σειριακές θύρες:** Το UML έχει μια κεντρική κονσόλα η οποία περιέχει τις εικονικές κονσόλες και τις σειριακές θύρες με ακριβή αντιστοιχία στη φυσική μηχανή. Για τη φυσική μηχανή αυτές οι οντότητες είναι διαφορετικές φυσικές συσκευές. Και οι δύο μπορούν να προσκολληθούν σε μια ποικιλία από συσκευές στο host, συμπεριλαμβανομένων των ttys, ptys, pts, xterms, υποδοχών (sockets) και στα ήδη υπάρχοντα file descriptors.
- **Block συσκευές:** Το UML έχει έναν block οδηγό, ο οποίος παρέχει πρόσβαση σε οτιδήποτε στο host, το οποίο μπορεί να γίνει mounted. Κανονικά χρησιμοποιείται για να κάνει mount συστήματα αρχείων από εικόνες στο σύστημα αρχείων του host. Μπορεί επίσης να χρησιμοποιηθεί για να παρέχει πρόσβαση σε συσκευές block στο host, όπως CD-ROMs, δισκέτες ή τμήματα σκληρού δίσκου.
- **Δικτυακές συσκευές:** Υπάρχει ένας οδηγός δικτύου, ο οποίος παρέχει στο UML δικτυακή πρόσβαση μέσω κάποιων μηχανισμών. Αυτοί περιλαμβάνουν slip, ethertap, TUN/TAP (network TUNnel / network TAP) και μια υποδοχή σε ένα daemon δρομολόγησης (routing daemon). Το slip είναι ικανό να ανταλλάσει IP πακέτα με το host και άλλες μηχανές με το host να παίζει το ρόλο του δρομολογητή. Τα ethertap και TUN/TAP ανταλλάσουν πλαίσια ethernet με το host και το εξωτερικό δίκτυο. Το daemon δρομολόγησης μπορεί να χρησιμοποιηθεί για να εγκαθυδρίσει ένα πλήρως εικονικό ethernet χωρίς καμιά σύνδεση με το host ή το φυσικό δίκτυο, αλλά και ένα εικονικό δίκτυο το οποίο είναι συνδεδεμένο με το φυσικό διαμέσου του daemon.

3.3.2 Σχεδιασμός και υλοποίηση

Kernel mode και user mode: Το UML πρέπει να παρέχει στον αρχικό πυρήνα όλες τις υποδομές ώστε αυτός να μπορεί να λειτουργήσει. Βασικό μέλημα είναι ο διαχωρισμός μεταξύ ενός μη εξουσιοδοτημένου χρήστη και ενός εξουσιοδοτημένου στον πυρήνα. Οι πλατφόρμες υλικού παρέχουν έναν ενσωματωμένο μηχανισμό για εναλλαγή μεταξύ αυτών των δύο και επιβάλλουν την έλλειψη εξουσιοδότησης στον χρήστη. Παρόλα αυτά το Linux δεν παρέχει τέτοιο μηχανισμό στις διεργασίες, έτσι το UML τον κατασκευάζει χρησιμοποιώντας την κλήση συστήματος ptrace. Το UML έχει ένα ειδικό νήμα του οποίου η κύρια δουλειά είναι να κάνει ptrace σε όλα τα άλλα νήματα. Σε ptrace υποβάλλονται μόνο οι διεργασίες που αφορούν το χρήστη, κι όχι αυτές που αφορούν τον πυρήνα. Αυτή είναι η διαφοροποίηση του kernel mode από το user mode.

Εικονικοποίηση κλήσεων συστήματος: Μέσω του μηχανισμού εναλλαγής από kernel mode σε user mode και αντίστροφα, η εικονικοποίηση των κλήσεων συστήματος είναι αρκετά απλή. Η κλήση συστήματος φυσικά θα πρέπει να ακυρωθεί στον πυρήνα του host. Αυτό γίνεται με το να αλλάξει ο καταχωρητής που περιέχει τον αριθμό της κλήσης συστήματος σε _NR_getpid. Όταν γίνει αυτό, το νήμα ptrace αποθηκεύει τους καταχωρητές διεργασίας στη δομή του νήματος και επαναφέρει μια κατάσταση που είχε αποθηκευθεί προηγουμένως. Η νέα κατάσταση αναγκάζει τη διεργασία να εκτελέσει τον διαχειριστή κλήσεων συστήματος (system call handler), ο οποίος διαβάζει τα γνωρίσματά της και φέρνει τον κώδικα της κλήσης συστήματος. Όταν η κλήση συστήματος τελειώσει, η διεργασία αποθηκεύει την τιμή που επέστρεψε στους καταχωρητές και κάνει αίτηση στο νήμα να επιστρέψει στο user mode. Το νήμα επαναφέρει το περιεχόμενο των καταχωρητών και συνεχίζει τη διεργασία.

Traps και faults: Ένα trap του επεξεργαστή είναι ένας άλλος μηχανισμός ώστε η διεργασία να μπει στον πυρήνα. Στο UML υλοποιείται με Linux σήματα. Όταν μια διεργασία λαμβάνει ένα σήμα, το νήμα ptrace το βλέπει πριν από αυτή. Τότε η διεργασία συνεχίζει στο kernel mode, αλλά χωρίς να αποθηκεύσει την κατάστασή της ή να επαναφέρει μια κατάσταση.

Διακοπές συσκευών: Οι διακοπές εξωτερικών συσκευών υλοποιούνται με το SIGIO. Οι οδηγοί ρυθμίζουν ότι όποτε φθάνουν δεδομένα εισόδου παράγεται ένα SIGIO. Ο SIGIO

handler καταλαβαίνει ποιοι file descriptors περιμένουν είσοδο και έτσι καθορίζει ποιο IRQ (Interrupt ReQuest) σχετίζεται με κάθε descriptor. Σε αυτό το σημείο καλεί τον τυπικό κώδικα IRQ για να χειριστεί τη διακοπή.

Διακοπές χρονομετρητών: Το ρολόι υλοποιείται με χρονομετρητές Linux. Αυτοί αποστέλλουν SIGALRM ή SIGVTALRM, ανάλογα με το αν η διεργασία που διακόπτεται είναι ένα ανενεργό νήμα ή όχι. Η διαφορά είναι ότι, αντίθετα με τα υπόλοιπα, το ανενεργό νήμα κάνει sleep και πρέπει να λάβει ένα SIGALRM.

Σφάλματα μνήμης: Τα σφάλματα μνήμης υλοποιούνται με το SIGSEGV. Όταν μια διεργασία UML κάνει μια μη έγκυρη πρόσβαση στη μνήμη ο host θα δημιουργήσει ένα SIGSEGV για αυτή. Ο SIGSEGV handler θα αποφασίσει αν η πρόσβαση είναι επιτρεπτή και απέτυχε επειδή η σελίδα δεν ειχε ακόμα αντιστοιχιθεί στη διεργασία ή αν η πρόσβαση δεν ήταν επιτρεπόμενη.

Context switching: Κάθε διεργασία UML έχει το δικό της νήμα στον πυρήνα του host. Έτσι το context switch από τη μία στην άλλη απαιτεί η πρώτη να σταματήσει και να συνεχίσει η επόμενη. Το UML επίσης εκχωρεί σε καθεμιά από τις διεργασίες το δικό της χώρο διευθύνσεων. Αυτό επιταχύνει το context switching. Όταν λαμβάνει χώρα ένα context switch ένα SIGIO σε ένα file descriptor θα πρέπει να αλλάξει από την παλαιά στη νέα διεργασία.

Εξομείωση εικονικής μνήμης: Το Linux διαθέτει πρόσβαση στη φυσική μνήμη, στην εικονική μνήμη του πυρήνα και στον εικονικό χώρο διευθύνσεων κάθε διεργασίας. Το UML το υλοποιεί δημιουργώντας ένα αρχείο μεγέθους φυσικής μνήμης και το απεικονίζει ως ένα block στο χώρο διευθύνσεων. Αυτό παρέχει την περιοχή φυσικής μνήμης του virtual machine. Η εικονική μνήμη του πυρήνα και των διεργασιών υλοποιείται με την απεικόνιση μεμονωμένων σελίδων από αυτό το αρχείο σε κατάλληλο τμήμα των εικονικών χώρων διευθύνσεων.

Πρόσβαση στο σύστημα αρχείων του host: Το UML έχει ένα εικονικό σύστημα αρχείων, το hostfs, το οποίο παρέχει απευθείας πρόσβαση στο σύστημα αρχείων του host. Αυτό γίνεται υλοποιώντας τη διεπαφή VFS. Οι περισσότερες λειτουργίες VFS μεταφράζονται απευθείας σε αντίστοιχες κλήσεις libc στο host.

Πολλαπλά περιβάλλοντα: Υπάρχει ένας αριθμός από εργαλεία και συστήματα τα οποία απαιτούν τη διατήρηση μιας ξεχωριστής μηχανής. Χαρακτηριστικά παραδείγματα είναι οι διαδικασίες εγκατάστασης, οι διαχειριστές πακέτων και οι δικτυακές υπηρεσίες. Εκείνοι που συντηρούν ή αναπτύσσουν πολλαπλές εκδόσεις αυτών των εργαλείων διατηρούν μια φυσική μηχανή για κάθε έκδοση. Το UML προσφέρει τη δυνατότητα να μετακινηθεί όλη αυτή η δραστηριότητα σε μια και μοναδική μηχανή, βάζοντας κάθε διαφορετική έκδοση σε μια ξεχωριστή εικονική μηχανή. Πέρα από το γεγονός ότι δεν απαιτείται συντήρηση πολλών φυσικών μηχανών, η συντήρηση των εικονικών είναι πιο βολική. Η ανάθεση πόρων μεταξύ των εικονικών μηχανών είναι πιο ευέλικτη, είναι πολύ πιο γρήγορες στην εκκίνηση και τον τερματισμό τους και μπορούν να δημιουργηθούν ή να καταργηθούν κατά βούληση. Είναι λοιπόν πιο εύκολο η όλη διαδικασία να γίνει εσωτερικά σε πολλαπλά UML παρά σε ξεχωριστές φυσικές μηχανές.

3.4 Linux Vserver

Το Linux VServer [19] είναι ένα σύστημα εικονικοποίησης που χρησιμοποιείται για να δημιουργήσει πολλά ανεξάρτητα containers σε ένα κοινό πυρήνα Linux. Τα containers φαίνονται σαν ξεχωριστά hosts τόσο στις εφαρμογές όσο και στους χρήστες. Η προσέγγιση του VServer, όπως θα δούμε βασίζεται στην έννοια του context isolation.

Ο πυρήνας έχει αλλαχθεί με τέτοιο τρόπο ώστε να απομονώνεται το container, δηλαδή αυτό δε μπορεί να επηρεάσει διεργασίες, αρχεία, δικτυακή κίνηση κλπ. που ανήκουν σε άλλα containers. Το Linux-VServer έχει αναπτυχθεί πριν αρκετά χρόνια και ο κύριος σχεδιαστικός του στόχος είναι να επιφέρει πολύ μικρή επιβάρυνση (overhead), ενώ ταυτόχρονα να αποτελεί μια αρκετά ποιοτική και ευέλικτη λύση. Χρησιμοποιείται κυρίως σε περιπτώσεις που απαιτούν μεγάλο βαθμό απομόνωσης σε συστήματα όπου η απόδοση είναι πολύ σημαντική, όπως web hosting συστήματα, υψηλής απόδοσης clusters και ενσωματωμένα συστήματα και για σταθεροποίηση server. Εναλλακτικές προσεγγίσεις στην υλοποίηση containers πάνω σε Linux περιλαμβάνουν το OpenVZ, το οποίο θα εξετάσουμε παρακάτω, το Virtuozzo, το Ensim. Όλες αυτές οι προσεγγίσεις επικεντρώνονται στο ίδιο στόχο, δηλαδή στην εικονικοποίηση συστήματος, αλλά διαφέρουν στην υλοποίηση.

3.4.1 Σχεδιασμός συστήματος

Σε υψηλό επίπεδο ένα σύστημα βασισμένο σε container παρέχει μια διαμοιραζόμενη, εικονικοποιημένη εικόνα λειτουργικού συστήματος, περιλαμβάνοντας ένα μοναδικό σύστημα αρχείων, ένα σύνολο από εκτελέσιμα συστήματος και βιβλιοθήκες και πόρους (κεντρική μονάδα επεξεργασίας, μνήμη, χώρους αποθήκευσης κλπ.) που αποδίδονται στο container όταν αυτό δημιουργείται. Κάθε container μπορεί να εκκινείται, να τερματίζει, όπως ένα κανονικό λειτουργικό σύστημα και να επανεκκινείται όποτε είναι απαραίτητο σε λίγα δευτερόλεπτα. Οι χρήστες και οι εφαρμογές έχουν την εντύπωση ότι δουλεύουν με ένα ξεχωριστό σύστημα Linux. Η πλατφόρμα που εμπεριέχει τα containers αποτελείται κυρίως από την διαμοιραζόμενη εικόνα του λειτουργικού συστήματος και από εξουσιοδοτημένη πληροφορία. Αυτή τη πληροφορία χρησιμοποιεί ο διαχειριστής συστήματος για να διαχειριστεί τα containers. Η εικονική πλατφόρμα είναι η όψη του συστήματος την οποία βλέπουν τα άλλα guest containers. Οι εφαρμογές που τρέχουν σε ένα guest container δουλεύουν ακριβώς με τον ίδιο τρόπο σαν να αντιστοιχούσαν σε σύστημα που δεν είναι βασισμένο σε αυτά και πιο συγκεκριμένα έχουν το δικό τους σύστημα αρχείων, τις δικές τους IP διευθύνσεις κλπ.

3.4.2 Επεκτάσεις στον Linux Kernel

Οι επεκτάσεις αυτές είναι σχεδιασμένες να έχουν μικρή επιβάρυνση, να είναι ευέλικτες και να βελτιώνουν την ασφάλεια του συστήματος με στόχο το σωστό «εγκλεισμό» των εφαρμογών σε ένα container. Όσον αφορά το χρονοπρογραμματισμό της κεντρικής μονάδας επεξεργασίας, το VServer παρουσιάζει μια καινοτόμο τεχνική φιλτραρίσματος, ώστε να υποστηρίξει ισομοιρασμό πόρων, διατήρηση εργασίας ή περιορισμένο χρονοπρογραμματισμό των containers. Παρόλα αυτά, υπό το πρίσμα της διαχείρισης των πόρων του συστήματος (π.χ. αποθηκευτικός χώρος, εύρος ζώνης εισόδου / εξόδου), για το VServer είναι κυρίως ζήτημα του να εξισορροπηθεί η υπάρχουσα διαχείριση πόρων. Τα συστήματα αρχείων των guest containers θα μπορούσαν είτε να υλοποιηθούν χρησιμοποιώντας loop-back mounted εικόνες (τυπικό για τα συστήματα qemu, xen, vmware και τα συστήματα βασισμένα σε uml) είτε απλώς χρησιμοποιώντας το πρωταρχικό σύστημα αρχείων και το chroot.

Το βασικό όφελος από τη χρήση ενός chroot-ed συστήματος αρχείων έναντι μιας loop-back mounted εικόνας είναι η απόδοση. Όμως υπάρχουν δύο μειονεκτήματα: α) η chroot() πληροφορία είναι αρκετά «ευμετάβλητη» ώστε να μπορεί να περιορισθεί σε container και β) τα chroot-ed συστήματα αρχείων μπορεί να οδηγήσουν σε μεγάλου πλήθους διπλότυπα των κοινών αρχείων. Το VServer αντιμετωπίζει και τα δύο αυτά προβλήματα, αφού ένας από τους βασικούς του στόχους είναι να υποστηρίξει τα containers με τόσο υψηλή απόδοση, ώστε να πλησιάζει κατά πολύ αυτή μιας μεμονωμένης διανομής Linux.

3.4.3 Όριο συστήματος αρχείων Chroot

Η chroot() πληροφορία, όπως προαναφέραμε είναι «ευμετάβλητη», γεγονός που δεν είναι επιτρεπτό από τη σκοπιά της ασφάλειας, όταν κάποιος θέλει να διατηρήσει αμετάβλητο το γεγονός ότι οι διεργασίες είναι εγκλεισμένες στο σύστημα αρχείων των containers. Για να κατανοήσουμε πόσο εύκολη είναι η διαφυγή από το chroot(), ας θεωρήσουμε τα ακόλουθα τρία βήματα: α) δημιουργία ή άνοιγμα ενός αρχείου και διατήρηση του file-descriptor, β) chroot σε ένα υποκατάλογο στο ίδιο ή χαμηλότερο επίπεδο αναφορικά με το αρχείο, το οποίο καθιστά τη ρίζα να κινηθεί πιο κάτω στο σύστημα αρχείων και γ) χρήση του fchdir() στο file descriptor για διαφυγή από τη νέα ρίζα, το οποίο έχει σαν αποτέλεσμα η διεργασία να διαφύγει από την παλιά ρίζα αφού αυτή χάθηκε στην τελευταία κλήση συστήματος chroot(). Για να αντιμετωπιστεί αυτό το πρόβλημα το VServer χρησιμοποιεί μια ειδική ιδιότητα αρχείου που ονομάζεται όριο chroot (chroot barrier). Το παραπάνω σενάριο δεν λειτουργεί όταν το όριο τεθεί στο γονικό κατάλογο ενός chroot-ed συστήματος αρχείων, αφού αποτρέπει μη εξουσιοδοτημένη τροποποίηση.

3.4.4 Ενοποίηση συστήματος αρχείων

Για να γίνει περισσότερο αντιληπτό το δεύτερο μειονέκτημα που αναφέρθηκε σε προηγούμενη παράγραφο, ας θεωρήσουμε ότι συστήματα με εκατοντάδες containers βασισμένη στην ίδια διανομή Linux θα διπλοεγγράψουν πολλά κοινά αρχεία. Αυτό συμβαίνει για να διατηρηθεί ένας βαθμός διαχωρισμού μεταξύ των containers, αφού θα ήταν δύσκολο να εξασφαλιστεί ο ασφαλής διαμοιρασμός αρχείων μεταξύ των containers με τη χρήση τεχνικών κλασσικού συστήματος αρχείων. Για την αντιμετώπιση αυτού του προβλήματος το VServer υλοποιεί μια τεχνική εξοικονόμησης χώρου στο δίσκο

χρησιμοποιώντας μια τεχνική ενοποίησης πάνω σε ολόκληρα αρχεία. Η βασική ιδέα είναι ότι τα αρχεία που είναι κοινά σε περισσότερα από ένα containers, τα οποία είναι σπάνιο να αλλάξουν (π.χ. βιβλιοθήκες και δυαδικά αρχεία από παρόμοιες διανομές λειτουργικού συστήματος), είναι hard-linked σε ένα διαμοιραζόμενο σύστημα αρχείων. Αυτό είναι πιθανό να συμβεί για τα guest containers μπορούν με ασφάλεια να διαμοιράζουν αντικείμενα του συστήματος αρχείων (inodes). Το μοναδικό μειονέκτημα με τα hard-linked αρχεία είναι ότι ένα container μπορεί να τροποποιήσει ή να διαγράψει τα κοινά αρχεία, το οποίο εν συνεχεία επηρεάζει άλλα containers. Αυτό μπορεί εύκολα να αντιμετωπιστεί δίνοντας μια αμετάβλητη ιδιότητα στο αρχείο, το οποίο μπορεί κατόπιν να διαμοιρασθεί σε δύο containers.

Από την άλλη μεριά, διαγράφοντας ή ενημερώνοντας ένα αρχείο με αμετάβλητο σύνολο ιδιοτήτων, που βρίσκεται μέσα σε ένα guest container θα ήταν αδύνατο. Άρα θα πρέπει να ορισθεί μια πρόσθετη ιδιότητα που θα αναιρεί τον παραπάνω περιορισμό. Αυτό αρκεί ώστε μια εφαρμογή που τρέχει μέσα σε ένα container να μπορεί να εκτελέσει ένα copy-on-write (CoW) με το να αντιγράψει το αρχικό αρχείο, να τροποποιήσει το αντίγραφο, να αποσυνδέσει το αρχικό αρχείο και τέλος να μετονομάσει το αντίγραφο στο όνομα του αρχικού αρχείου. Αυτή η τεχνική χρησιμοποιήθηκε από παλιότερα VServer συστήματα, αλλά δημιουργούσε προβλήματα ασυμβατότητας σε προγράμματα που τροποποιούν αρχεία. Για να αντιμετωπιστεί αυτό, το VServer υιοθετεί ένα αποσυνδεδεμένο CoW, το οποίο χειρίζεται τα διαμοιραζόμενα hard-linked αρχεία σαν υποψήφια για CoW. Όταν ένα container προσπαθήσει να αλλάξει ένα αρχείο μαρκαρισμένο ως CoW, ο πυρήνας θα δημιουργήσει ένα ιδιωτικό αντίγραφο του αρχείου για το container. Τέτοια αρχεία μαρκαρισμένα ως CoW, που ανήκουν σε περισσότερα από container ονομάζονται «ενοποιημένα» (unified) και γη διεργασία της εύρεσης κοινών αρχείων και της επακόλουθης διαχείρισής των ονομάζεται ενοποίηση συστήματος αρχείων (Filesystem Unification). Ο πρωταρχικός στόχος για την ενοποίηση συστήματος αρχείων είναι η μειωμένη κατανάλωση πόρων. Ενώ μια τυπική διανομή Linux εγκαθιστά περίπου 500MB χώρο στο δίσκο, μετά από ενοποίηση η δημιουργία ενός νέου container βασισμένου στην ίδια διανομή απαιτεί μόνο λίγα MBs.

3.4.5 Απομόνωση διεργασιών

Το VServer χρησιμοποιεί τον καθολικό χώρο PID για όλα τα containers. Ο σκοπός είναι να κρύψει όλες τις διεργασίες που δεν σχετίζονται με το container και να αποτρέψει οποιαδήποτε ανεπιθύμητη αλληλεπίδραση μεταξύ διεργασιών που βρίσκονται σε διιαφορετικά containers. Αυτός ο διαχωρισμός απαιτεί μια επέκταση μερικών ήδη υπάρχουσων δομών δεδομένων στον πυρήνα ώστε αυτές να α) μάθουν σε ποιό container ανήκουν, β) διαφοροποιηθούν ανάμεσα σε ίδια UIDs που χρησιμοποιούνται από διαφορετικά containers. Όταν ένα VServer σύστημα εκκινείται, εξ ορισμού όλες οι διεργασίες ανήκουν στο host. Για να απλοποιηθεί η διαχείριση συστήματος ο host συμπεριφέρεται ως ένα κανονικό σύστημα Linux και δεν εκθέτει πληροφορία για τους guests. Παρόλα αυτά για να επιτραπεί μια καθολική επισκόπηση των διεργασιών, το VServer καθορίζει έναν ειδικό «παρατηρητή» ο οποίος μπορεί να επιτηρεί ταυτόχρονα όλες τις διεργασίες. Τόσο ο host όσο και ο παρατηρητής είναι «λογικά» containers (δεν υλοποιούνται από δομές δεδομένων του πυρήνα). Μια παρενέργεια αυτής της προσέγγισης είναι οτι το migration διεργαζιών από ένα container σε ένα άλλο στο ιδιο host επιτυγχάνεται με την αλλαγή της συσχέτισης του container και την ενημέρωση των αντίστοιχων στατιστικών που αφορούν τη χρήση των πόρων για κάθε container, όπως NPROC, NOFILE, RSS, ANON, MEMLOCK. Το όφελος από αυτή την απομόνωση διεργασιών είναι α) είναι αρκετά επεκτάσιμο, β) οι πιο κρίσιμες στη διαχείριση διεργασίες και τα PIDs παραμένουν αμετάβλητα. Το μειονέκτημα είναι ότι κάποιος δεν μπορεί εύκολα να κάνει migration του container, να κρατήσει σημείο αναφοράς και να συνεχίσει από αύτο, γιατί δεν είναι σίγουρο ότι οι διεργασίες θα λάβουν το ίδιο PID που είχαν πριν το migration. Για να αποφευχθεί αυτό, εναλλακτικά container-based συστήματα εικονικοποιούν τον χώρο των PID για κάθε container.

3.4.6 Δικτυακή απομόνωση

Το VServer δεν εικονοποιεί πλήρως το δικτυακό υποσύστημα. Διαμοιράζει π.χ. τους πίνακες δρομολόγησης ή τους πίνακες IP διευθύνσεων μεταξύ όλων των containers, αλλά τα περιορίζει να αντιστοιχούν υποδοχές (sockets) σε ένα υποσύνολο από IPs του host που καθορίζονται είτε κατά τη δημιουργία του container είτε δυναμικά από το διαχειριστή του

host. Αυτό έχει σαν μειονέκτημα ότι δεν αφήνει τα containers να αλλάξουν τους πίνακες δρομολόγησής τους. Με αυτόν τον τρόπο όμως τα containers του VServer επιτυγχάνουν αρκετά υψηλή δικτυακή απόδοση. Αναφορικά με τη δικτυακή απομόνωση του VServer θα πρέπει να ληφθεί υπόψη το γεγονός ότι αντιστοιχήσεις με ειδικές διευθύνσεις όπως το IPADDR_ANY ή με τη διεύθυνση localhost, θα πρέπει να αντιμετωπιστούν ώστε να αποτραπεί σε ένα container να λαμβάνει δικτυακή κίνηση που ανήκει σε ένα άλλο. Η βασική προσέγγιση για να αποτραπούν τέτοιες καταστάσεις περιλαμβάνει το tagging στα πακέτα packets με το κατάλληλο identifier του container και η ενσωμάτωση των κατάλληλων φίλτρων στη στοίβα δικτύου ώστε να εξασφαλιστεί ότι θα τα λάβει μόνο το σωστό container. Συγκριτικές μετρήσεις επιδόσεων έδειξαν ότι η επιβάρυνση που εγείρεται είναι μηδαμινή καθώς η δικτυακή απόδοση σε υψηλές ταχύτητες δεν διαφοροποιείται ανάμεσα σε μεμονωμένα Linux συστήματα και σε VServer συστήματα και δεν επηρεάζεται από το πλήθος των ταυτόχρονα ενεργών containers. Στον αντίποδα, η καλύτερη εικονοποίηση δικτύου L2 ή L3 όπως υλοποιείται σε διάφορα συστήματα με containers επιφέρουν σημαντική επιβάυνση στη κεντρική μονάδα επεξεργασίας όταν αυξάνεται ο αριθμός των containers στο σύστημα. Βέβαια δε θα πρέπει να ξεχνάμε ότι παρόλο που η εικονοποίηση δικτύου είναι αρκετά ευέλικτη, δεν ενδείκνυται σε όλες τις περιπτώσεις.

3.4.7 Απομόνωση CPU

Το VServer υλοποιεί την απομόνωση CPU με επικάλυψη ενός σχήματος κουβά κουπονιών (token bucket) σε έναν τυπικό χρονοπρογραμματιστή CPU του Linux. Κάθε container έχει ένα token bucket το οποίο συσσωρεύει τα tokens με ένα συγκεκριμένο ρυθμό. Σε κάθε χτύπο του ρολογιού, το container στο οποίο ανήκει η τρέχουσα διεργασία χρεώνεται ένα κουπόνι. Το container που μείνει χωρίς κουπόνια αδειάζει την ουρά διεργασιών προς εκτέλεση έως ότου ο δικός του bucket συγκεντρώσει ένα ελάχιστο αριθμό κουπονιών. Αυτό το σχήμα κουπονιών μπορεί να χρησιμοποιηθεί για να παρέχει έναν δίκαιο διαμοιρασμό της CPU στις διεργασίες. Μπορεί επίσης να επιβάλλει ένα άνω όριο στον αριθμό των κύκλων της CPU που ένα container μπορεί να χρησιμοποιήσει ακόμα και στην περίπτωση που το σύστημα παραμένει ανενεργό.

Ο ρυθμός συσσώρευσης των κουπονιών στο bucket του container εξαρτάται από το αν το container έχει κράτηση ή διαμοιράζει. Ένα container με κράτηση συσσωρεύει κουπόνια με το ρυθμό που κάνει κράτηση, για παράδειγμα ένα container με 10% κράτηση λαμβάνει 100

κουπόνια το δευτερόλεπτο, μιας και το κουπόνι του δίνει το δικαίωμα να τρέξει μια διεργασία για ένα millisecond. Ένα container που διαμοιράζει και έχει διεργασίες προς εκτέλεση θα προγραμματιστεί μόνο όταν όλα τα containers με κρατήσεις έχουν ικανοποιηθεί. Το αποτέλεσμα είναι ότι η CPU εκχωρείται αποδοτικά και στις δύο περιπτώσεις: τα containers με κρατήσεις χρησιμοποιούν τη CPU ανάλογα με τις κρατήσεις και τα containers που διαμοιράζουν χρησιμοποιούν το χρόνο που η CPU μένει ανενεργή κατά αντιστοιχία.

3.4.8 QoS δικτύου

Το Hierarchical Token Bucket (HTB) στον έλεγχο κίνησης (Traffic Control) του Linux μπορεί να χρησιμοποιηθεί για να παρέχει το επιθυμητό εύρος ζώνης σε μια δίκαιη υπηρεσία. Για τα containers τα οποία έχουν τις δικές τους IP διευθύνσεις, η υποστήριξη του πυρήνα στο HTB δεν απαιτεί τροποποιήσεις. Παρόλα αυτά όταν τα containers διαμοιράζουν μια διεύθυνση IP, όπως γίνεται στο PlanetLab, είναι απαραίτητο να ανιχνεύονται τα πακέτα ώστε να τηρείται ενας κανόνας στην δικτυακή κίνηση containers. Αυτό επιτυγχάνεται με το να γίνεται tagging στα πακέτα που αποστέλλει ένα container με το δικό του id στον πυρήνα. Έπειτα δημιουργείται για κάθε container ένα bucket κουπονιών με ρυθμό κρατήσεων και διαμοιρασμό. Το πρώτο δείχνει το μέγεθος του εξερχομενου εύρους ζώνης που εκχωρείται στο container και το δεύτερο καθορίζει τον τρόπο με τον οποίο το container μοιράζει το εύρος ζώνης ξέχωρα από την κράτησή του. Εν συνεχεία το HTB επιτρέπει σε κάθε container να στέλνει πακέτα στο ρυθμό κράτησης του bucket κουπονιών του και διανέμει το περίσσευμα σε εύρος ζώνης στα άλλα containers αναλογικά με αυτά που διαμοιράζουν.
Το VServer διαχερίζεται το I/O του δίσκου χρησιμοποιώντας το τυπικό για το Linux CFQ (Completely Fair Queuing) I/O χρονοπρογραμματιστή. Ο CFQ χρονοπρογραμματιστής προσπαθεί να χωρίσει το εύρος ζώνης σε κάθε συσκευή με δίκαιο τρόπο ανάμεσα στα containers που εκτελούν I/O σε αυτή τη συσκευή.

3.5 OPENVZ

Το OpenVZ είναι μια υλοποίηση containers στο Linux. Αποτελείται από έναν τροποποιημένο πυρήνα Linux και εργαλεία χρήστη. Ο τροποποιημένος πυρήνας Linux του

OpenVZ προσθέτει την ακόλουθη λειτουργικότητα: εικονοποίηση και απομόνωση διάφορων υποσυστημάτων, διαχείριση πόρων και σημεία ελέγχου (checkpoints). Η εικονικοποίηση και απομόνωση δημιουργεί πολλά εικονικά περιβάλλοντα μέσα σε έναν και μοναδικό πυρήνα. Το υποσύστημα διαχείρισης πόρων περιορίζει και σε κάποιες περιπτώσεις εξασφαλίζει πόρους όπως CPU, μνήμη και χώρο στο δίσκο σε κάθε εικονικό περιβάλλον. Η δημιουργία σημείων ελέγχου είναι μια διαδικασία κατά την οποία ολόκληρη η κατάσταση ενός εικονικού περιβάλλοντος αποθηκεύεται σε ένα αρχείο στο δίσκο, με την ικανότητα αυτή η κατάσταση να ανακτηθεί κάποια μεταγενέστερη χρονική στιγμή.

3.5.1 Εικονοποίηση και απομόνωση

Κάθε εικονικό περιβάλλον έχει το δικό του σύνολο πόρων που του έχει ανατεθεί από τον πυρήνα του λειτουργικού συστήματος. Μέσα στον πυρήνα, οι πόροι αυτοί έχουν είτε εικονοποιηθεί είτε απομονωθεί. Κάθε εικονικό περιβάλλον έχει επίσης το δικό του σύνολο από αντικείμενα:

Αρχεία: βιβλιοθήκες συστήματος, εφαρμογές, εικονοποιημένα /proc και /sys, κλπ.

Χρήστες και ομάδες χρηστών: κάθε εικονικό περιβάλλον έχει το δικό του root χρήστη καθώς και άλλους χρήστες και ομάδες χρηστών.

Δέντρο διεργασιών: ένα εικονικό περιβάλλον βλέπει μόνο το δικό του σύνολο διεργασιών και πρώτη όλων την init. Τα PIDs εικονοποιούνται έτσι ώστε το PID της init να είναι 1.

Δίκτυο: συσκεύη εικονικού δικτύου, η οποία επιτρέπει στο εικονικό περιβάλλον να έχει τη δική του IP διεύθυνση, καθώς και ένα σύνολο από πίνακες IP και κανόνες δρομολόγησης.

Συσκευές: κάποιες συσκευές εικονικοποιούνται. Επιπρόσθετα, σε κάθε εικονικό περιβάλλον μπορεί να αποδοθεί πρόσβαση σε φυσικές συσκευές όπως διεπαφές δικτύου, σειριακές θύρες, σκληρούς δίσκους κλπ.

IPC αντικείμενα: διαμοιραζόμενη μνήμη, σηματοφορείς και μηνύματα.

3.5.2 Διαχείριση πόρων

Η διαχείριση πόρων είναι κυρίαρχης σημασίας για λύσεις εικονικοποίησης λειτουργικού συστήματος, διότι υπάρχει ένα πεπερασμένο σύνολο πόρων μέσα σε ένα και μοναδικό πυρήνα, το οποίο διαμοιράζεται σε πολλά εικονικά περιβάλλοντα. Όλοι οι πόροι θα πρέπει

να ελέγχονται με τέτοιο τρόπο ώστε πολλά εικονικά περιβάλλοντα να συνυπάρχουν σε ένα μεμονωμένο σύστημα χωρίς να επηρεάζουν το ένα το άλλο. Το υποσύστημα διαχείρισης πόρων του OpenVZ αποτελείται από τρία στοιχεία:

1. **Διεπίπεδη ποσόστωση δίσκου (Two-level disk quota):** Ο διαχειριστής του OpenVZ εξυπηρέτη μπορεί να εγκαταστήσει quotas δίσκου για κάθε εικονικό περιβάλλον με βάση το χώρο του δίσκου και τον αριθμό των inodes. Αυτό ειναι το πρώτο επίπεδο ποσόστωσης. Στο δεύτερο επίπεδο ο διαχειριστής του εικονικού περιβάλλοντος (ή ο root χρήστης) χρησιμοποιεί τυπικά εργαλεία του UNIX για εγκαταστήσει quotas για κάθε χρήστη και κάθε ομάδα χρηστών.
2. **Δίκαιος χρονοπρογραμματιστής CPU:** Ο χρονοπρογραμματιστής CPU του OpenVZ είναι επίσης διεπίπεδος. Στο πρώτο επίπεδο αποφασίζει σε ποιο εικονικό περιβάλλον θα δώσει slice χρόνου, λαμβάνοντας υπόψιν την προτεραιότητα του εικονικού περιβάλλοντος για τη CPU. Στο δεύτερο επίπεδο, ο τυπικός χρονοπρογραμματιστής Linux αποφασίζει ποια διεργασία στο εικονικό περιβάλλον θα πάρει χρονικό slice, λαμβάνοντας υπόψιν τυπικές προτεραιότητες διεργασίας.
3. **Beancounters χρηστών:** Αυτό είναι ένα σύνολο, για κάθε εικονικό περιβάλλον, από μετρητές, περιορισμούς και εγγυήσεις. Υπάρχει ένα σύνολο από περίπου είκοσι παραμέτρους, οι οποίες έχουν επιλεχθεί για να καλύψουν όλες τις περιπτώσεις λειτουργίας των εικονικών περιβαλλόντων, έτσι ώστε κανένα από αυτά να κάνει κατάχρηση πόρων.

Οι πόροι που λογίζονται και ελέγχονται είναι κυρίως η μνήμη και διάφορα αντικείμενα στον πυρήνα, όπως τμήματα IPC διαμοιραζόμενης μνήμης, δικτυακές προσωρινές μνήμες (buffers), κλπ.

3.5.3 Δημιουργία Σημείων Ελέγχου (Checkpointing) - Migration

Η δημιουργία σημείων ελέγχου επιτρέπει το migration ενός εικονικού περιβάλλοντος σε κάποιον άλλο φυσικό εξυπηρέτη. Το εικονικό περιβάλλον «παγώνει» και ολόκληρη η κατάστασή του αποθηκεύεται σε αρχείο στο σκληρό δίσκο. Αυτό το αρχείο μπορεί έπειτα να μεταφερθεί σε άλλο μηχάνημα και να ανακτηθεί σε αυτό. Η όλη διαδικασία απαιτεί λίγα δευτερόλεπτα και από τη μεριά του client δεν φαίνεται σαν χρόνος διακοπής, αλλά σαν καθυστέρηση στην επεξεργασία, αφού και οι δικτυακές συνδέσεις υπόκεινται σε migration.

3.5.4 Πλεονεκτήματα χρήσης OpenVz

- **Σταθεροποίηση εξυπηρετητή (Server Consolidation):** Επιτρέπει σε έναν οργανισμό να μειώσει τον αριθμό των φυσικών εξυπηρετών μετακινώντας τις εφαρμογές τους σε εικονικά περιβάλλοντα. Ο αριθμός των λειτουργικών συστημάτων παραμένει ο ίδιος. Αυτό οδηγεί σε εξοικονόμηση κόστους σε υλικό, παροχή ενέργειας και διαχείριση.
- **Ασφάλεια:** Μπορεί να βελτιωθεί δραστικά θέτοντας κάθε δικτυακή υπηρεσία (π.χ. web server, mail server, κλπ.) σε ένα ξεχωριστό απομονωμένο εικονικό περιβάλλον. Σε περίπτωση κενού ασφαλείας στις εφαρμογές, τα υπόλοιπα περιβάλλοντα δεν επηρεάζονται. Επιπρόσθετο πλεονέκτημα θεωρείται η δυνατότητα δυναμικής διαχείρισης πόρων και το migration εφαρμογών σε πραγματικό χρόνο.
- **Hosting:** Η εικονικοποίηση σε επίπεδο λειτουργικού συστήματος είναι ο μόνος τύπος εικονικοποίησης τον οποίο οι πάροχοι υπηρεσιών μπορούν οικονομικά να υποστηρίξουν και να χρησιμοποιήσουν ώστε να πουλήσουν φθηνά εικονικά περιβάλλοντα στους πελάτες τους. Να σημειωθεί ότι κάθε εικονικό περιβάλλον έχει πλήρης root πρόσβαση, με την έννοια ότι κάθε ιδιοκτήτης εικονικού περιβάλλοντος μπορεί να επανεγκαταστήσει οτιδήποτε, ακόμα και να χρησιμοποιήσει πράγματα όπως πίνακες IP διευθύνσεων ή κανόνες τείχους προστασίας.
- **Ανάπτυξη λογισμικού και έλεγχος:** Συνήθως οι developers και οι testers χρειάζονται πρόσβαση σε διάφορες διανομές Linux και συχνά απαιτείται να τις επανεγκαταστήσουν. Με τη χρήση εικονικοποίησης οι developers μπορούν να χειρίζονται πολλές διανομές σε ένα μόνο εξυπηρετητή με υψηλή απόδοση χωρίς να επανεκκινούν το σύστημα. Η δημιουργία κλώνων εικονικών περιβάλλοντων είναι επίσης μια απλή διαδικασία.
- **Εκπαίδευση:** Κάθε σπουδαστής μπορεί να έχει το δικό του εικονικό περιβάλλον με διαφορετικά λειτουργικά συστήματα. Η δημιουργία ή επαναδημιουργία ενός εικονικού περιβάλλοντος είναι μια εύκολη διαδικασία.

3.6 OpenFlow

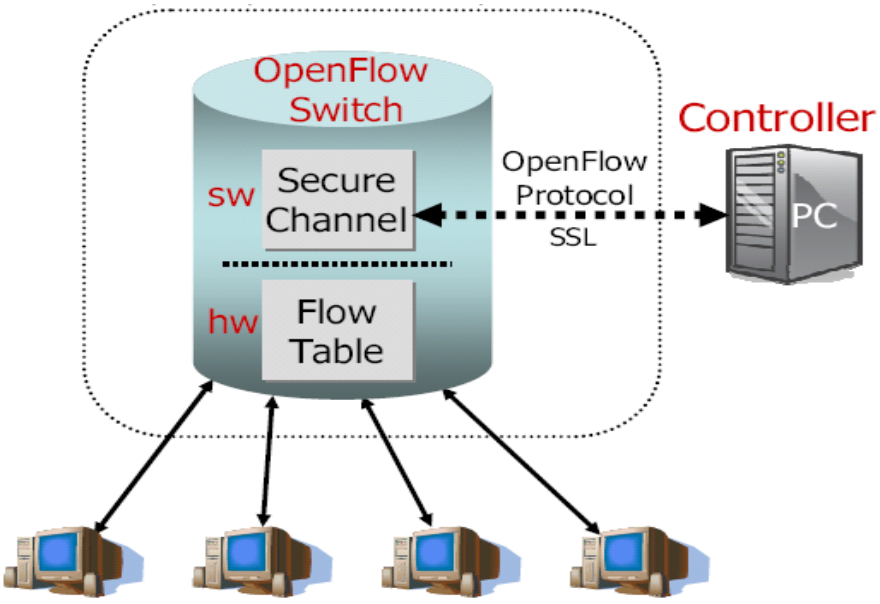
Οι περισσότεροι μεταγωγείς Ethernet και δρομολογητές περιέχουν πίνακες ροής (flow-tables), οι οποίοι τυπικά διαμορφώνονται από τα TCAMs (Ternary Content-Addressable Memory). Τα TCAMs υλοποιούν τείχη προστασίας, NAT (Network Address Translation), QoS (Quality of Service) και συλλέγουν στατιστικά δεδομένα. Ενώ ο πίνακας ροής κάθε

κατασκευαστή είναι διαφορετικός, ένα σύνολο από λειτουργίες είναι κοινό σε πολλούς μεταγωγείς και δρομολογητές. Το OpenFlow [17] εκμεταλλεύεται αυτό το κοινό σύνολο λειτουργιών και παρέχει ένα ανοιχτό πρωτόκολλο για τον προγραμματισμό του πίνακα ροής σε διαφορετικούς μεταγωγείς και δρομολογητές. Οι ερευνητές μπορούν να ελέγχουν τις ροές τους επιλέγοντας τις διαδρομές που θα ακολουθήσουν τα πακέτα τους και την επεξεργασία που αυτά δέχονται. Με αυτό τον τρόπο, οι ερευνητές μπορούν να δοκιμάσουν νέα πρωτόκολλα δρομολόγησης, μοντέλα ασφάλειας, σχήματα διευθυνσιοδότησης.

3.6.1 Αρχιτεκτονική OpenFlow

Το datapath του μεταγωγέα OpenFlow αποτελείται από ένα πίνακα ροής και μια ενέργεια που σχετίζεται με κάθε είσοδο ροής. Το σύνολο των ενεργειών που υποστηρίζεται από ένα μεταγωγέα OpenFlow επεκτείνεται συνεχώς. Παρακάτω θα περιγράψουμε τις ελάχιστες απαιτήσεις που αφορούν το σύνολο των μεταγωγέων. Για υψηλή απόδοση και χαμηλό κόστος το datapath πρέπει να έχει ένα προσεκτικά προκαθορισμένο βαθμό ευελιξίας. Αυτό σημαίνει αφενός την αυθαίρετη διαχείριση κάθε πακέτου, αφετέρου θέτει περιορισμό στην ποικιλία των ενεργειών.

Ένας OpenFlow μεταγωγέας αποτελείται από τουλάχιστον τρία τμήματα: α) ένα πίνακα ροής με μια ενέργεια σχετική με κάθε εγγραφή ροής, β) ένα ασφαλές κανάλι το οποίο συνδέει το μεταγωγέα με τη διεργασία απομακρυσμένου ελέγχου που ονομάζεται controller, επιτρέποντας έτσι εντολές και πακέτα να αποστέλλονται μεταξύ του controller και του μεταγωγέα χρησιμοποιώντας γ) το πρωτόκολλο OpenFlow, το οποίο παρέχει ένα ανοιχτό και πρότυπο τρόπο για τον controller να επικοινωνεί με τον μεταγωγέα.



Καθορίζοντας μια πρότυπη διεπαφή μέσω της οποίας οι εγγραφές στον πίνακα ροής μπορούν να καθοριστούν εξωτερικά, ο OpenFlow μεταγωγέας αποφεύγει την ανάγκη οι ερευνητές να τον προγραμματίσουν. Είναι χρήσιμο να κατηγοριοποιηθούν οι μεταγωγείς σε dedicated OpenFlow μεταγωγείς, οποίοι δεν υποστηρίζουν κανονική επεξεργασία Layer 2 και Layer 3 και εμπορικοί OpenFlow-enabled Ethernet μεταγωγείς γενικού σκοπού και δρομολογητές, στους οποίους έχει ενσωματωθεί το OpenFlow πρωτόκολλο και οι διεπαφές ως ένα καινούριο χαρακτηριστικό.

3.6.2 Dedicated OpenFlow μεταγωγείς

Ένας dedicated OpenFlow μεταγωγέας είναι ένα στοιχείο το οποίο προωθεί πακέτα μεταξύ θυρών, όπως καθορίζεται από μια διεργασία απομακρυσμένου ελέγχου. Κατά αυτή την έννοια, οι ροές καθορίζονται γενικά και περιορίζονται μόνο στις δυνατότητες της συγκεκριμένης υλοποίησης του πίνακα ροής. Για παράδειγμα, μια ροή μπορεί να ειναι μια σύνδεση TCP ή όλα τα πακέτα από μια συγκεκριμένη διεύθυνση MAC ή IP διεύθυνση ή όλα τα πακέτα με την ίδια ετικέτα VLAN ή όλα τα πακέτα από την ίδια θύρα μεταγωγέα. Για πειράματα που δεν περιλαμβάνουν πακέτα IPv4, μια ροή μπορεί να καθοριστεί ως όλα τα πακέτα που έχουν μια συγκεκριμένη επικεφαλίδα. Κάθε εγγραφή ροής έχει μια απλή ενέργεια που σχετίζεται με αυτή. Οι τρεις βασικές ενέργειες που όλοι οι dedicated OpenFlow μεταγωγείς πρέπει να υποστηρίζουν είναι:

1. Προώθηση των πακέτων της ροής σε μια δεδομένη θύρα (ή θύρες).

Αυτό επιτρέπει στα πακέτα να δρομολογηθούν στο δίκτυο. Στους περισσότερους μεταγωγείς αυτό αναμένεται να γίνεται με ταχύτητα όση η ταχύτητα της ζεύξης.

2. Ενθυλάκωση και προώθηση των πακέτων της ροής σε ένα controller.

Το πακέτο μεταφέρεται σε ένα ασφαλές κανάλι, όπου ενθυλακώνεται και αποστέλλεται στο controller. Τυπικά χρησιμοποιείται για το πρώτο πακέτο μιας νέας ροής, έτσι ο controller μπορεί να αποφασίσει αν η ροή θα προστεθεί στον πίνακα ροής. Σε κάποια πειράματα μπορεί να χρησιμοποιηθεί για να προωθήσει όλα τα πακέτα στο controller για επεξεργασία.

3. Απόρριψη των πακέτων της ροής.

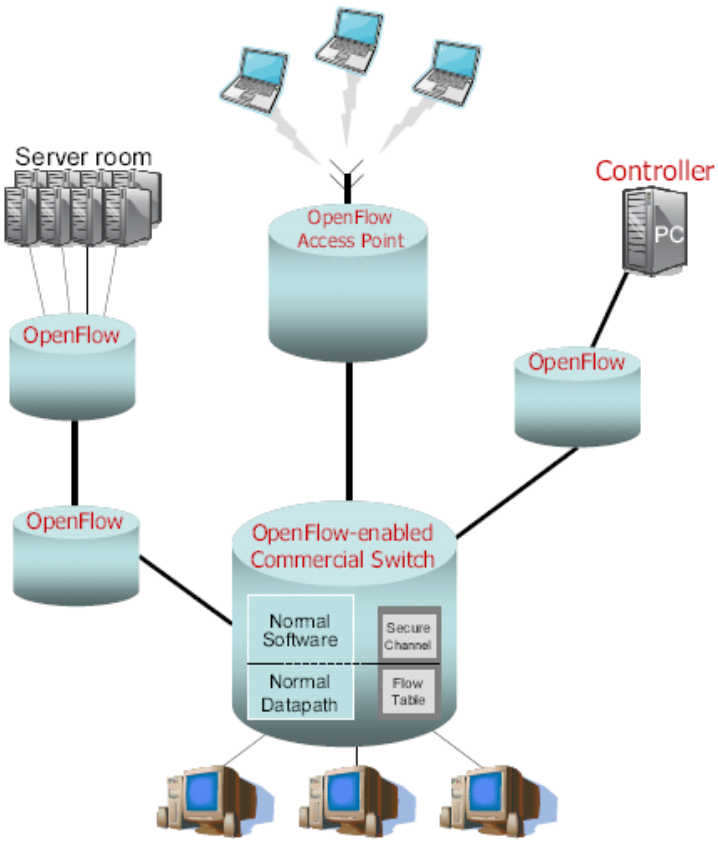
Μπορεί να χρησιμοποιηθεί για ασφάλεια, για περιορισμό επιθέσεων τύπου DoS (Denial of Service), ή να μειώσει ψεύτικη δικτυακή κίνηση από τους end-hosts.

Μια εγγραφή στο Flow-Table έχει τρία πεδία: α) μια επικεφαλίδα πακέτου που καθορίζει τη ροή, β) την ενέργεια, η οποία καθορίζει το ποια επεξεργασία θα γίνει στα πακέτα και γ)

στατιστικά, τα οποία σχετίζονται με τον αριθμό των πακέτων και των bytes για κάθε ροή και την ώρα του τελευταίου πακέτου της ροής. Με αυτόν τον τρόπο διαγράφονται ανενεργές ροές. Στην πρώτη γενιά των «Type 0» μεταγωγέων, η επικεφαλίδα ροής είναι μια δεκάδα. ,οα ροή TCP καθορίζεται και από τα δέκα πεδία, ενώ μια IP ροή μπορεί να μην περιέχει τις θύρες μεταφοράς στον προσδιορισμό της. Κάθε πεδίο επικεφαλίδας μπορεί να είναι wildcard για να επιτραπεί η συνάθροιση ροών, κατά αντίστοιχο τρόπο με ροές στις οποίες καθορίζεται μόνο το VLAN ID και αφορούν όλη την κίνηση σε ένα συγκεκριμένο VLAN.

3.6.3 OpenFlow-enabled μεταγωγείς

Κάποιοι εμπορικοί μεταγωγείς, δρομολογητές και access points αναβαθμίζονται με το OpenFlow προσθέτοντας σε αυτά τον πίνακα ροής, το ασφαλές κανάλι και το OpenFlow πρωτόκολλο. Τυπικά ο πίνακας ροής επαναχρησιμοποιεί το υπάρχον υλικό, όπως το TCAM, το ασφαλές κανάλι και το πρωτόκολλο τρέχουν στο λειτουργικό σύστημα του μεταγωγέα. Ένα δίκτυο από OpenFlow-enabled εμπορικούς μεταγωγείς access points φαίνεται στο παρακάτω σχήμα.



Όπως φαίνεται στο σχήμα, ο ίδιος controller διαχειρίζεται όλους τους πίνακες ροής. Το OpenFlow πρωτόκολλο επιτρέπει σε έναν μεταγωγέα να ελέγχεται από δύο ή περισσότερους controllers για αυξημένη απόδοση.

3.6.4 Κατηγοριοποίηση μεταγωγέων

Αν ένας μεταγωγέας υποστηρίζει τις μορφές επικεφαλίδας και τις βασικές ενέργειες που περιγράφηκαν προηγουμένως τότε ονομάζεται «Type 0» μεταγωγέας. Πολλοί μεταγωγείς ενδέχεται να υποστηρίζουν επιπλέον ενέργειες, για παράδειγμα να επανεγγράφουν τμήματα της επικεφαλίδας του πακέτου και να αντιστοιχίζουν πακέτα σε κλάσεις προτεραιότητας. Επίσης κάποιοι πίνακες ροής μπορεί να ταιριάζουν αυθαίρετα πεδία στην επικεφαλίδα του πακέτου, δίνοντας τη δυνατότητα χρήσης νέων πρωτοκόλλων που δεν βασίζονται στο IP. Τα καινούρια σύνολα χαρακτηριστικών θα χαρακτηρίσουν τους «Type 1» μεταγωγείς.

3.6.5 Controllers

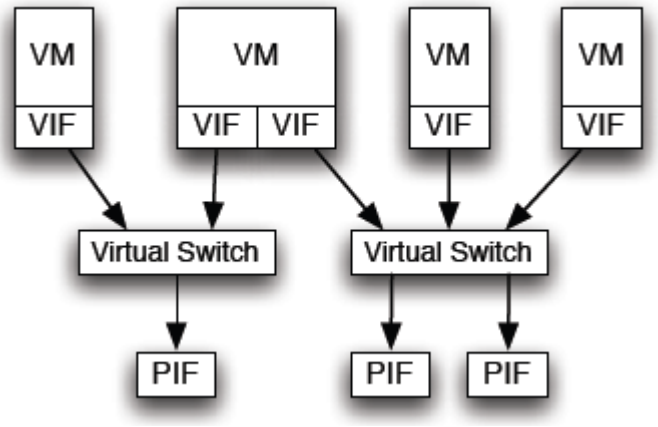
Ένας controller προσθέτει και διαγράφει εγγραφές ροών του πίνακα ροής. Για παράδειγμα ένας στατικός controller μπορεί να είναι μια απλή εφαρμογή σε ένα PC για να εγκαθιδρύει στατικά ροές διασύνδεσης με ένα σύνολο υποολογιστών κατα τη διάρκεια ενός πειράματος. Σε αυτή την περίπτωση οι ροές μοιάζουν με τα γνωστά VLANs στα υπάρχοντα δίκτυα. Από αυτή τη σκοπιά το OpenFlow είναι μια γενίκευση των VLANs. Πιο πολύπλοκοι controllers μπορούν δυναμικά να προσθέτουν και να διαγράφουν ροές. Σε ένα σενάριο ο ερευνητής μπορεί να ελέγξει ολόκληρο το δίκτυο από OpenFlow μεταγωγείς και να είναι ελεύθερος να αποφασίσει τον τρόπο επεξεργασίας των ροών. Ένας ακόμα πιο πολύπλοκος μπορεί να υποστηρίζει πολλούς λογαριασμούς χρηστών με διαφορετικές δυνατότητες, δίνοντας έτσι τη δυνατότητα να λαμβάνουν χώρα πολλά ανεξάρτητα πειράματα σε διαφορετικές ομάδες ροών. Οι ροές μπορούν να χαρακτηριστούν ως να είναι υπό τον έλεγχο ενός χρήστη, με τη βοήθεια ενός policy table που τρέχει στον controller. Ακολούθως μπορεί να μεταφερθεί στο πρόγραμμα επιπέδου χρήστη, το οποίο αποφασίζει εάν μια καινούρια εγγραφή ροής θα πρέπει να προστεθεί στο δίκτυο των μεταγωγέων.

3.7 Open vSwitch

Στο νέο διαμορφωμένο virtual περιβάλλον προκειμένου να διασφαλιστεί η διασύνδεση των virtual machines υλοποιήθηκε το Open vSwitch ένα virtual switch ειδικό για virtual περιβάλλοντα. Το vSwitch διαφέρει από άλλες υλοποιήσεις διότι διαθέτει μια εξωτερική διεπαφή για διαχειριστικούς λόγους, για διασφάλιση ποιότητας υπηρεσίας, υποστηρίζει tunneling καθώς και τήρηση συγκεκριμένων κανόνων φιλτραρίσματος των πακέτων. Το vSwitch με χρήση συγκεκριμένων μεθόδων που θα περιγραφούν παρακάτω χειρίζεται κρίσιμα ζητήματα όπως η απομόνωση και η μεταφορά virtual machines σε διαφορετικό subnet.

3.7.1 Αρχιτεκτονική Open vSwitch

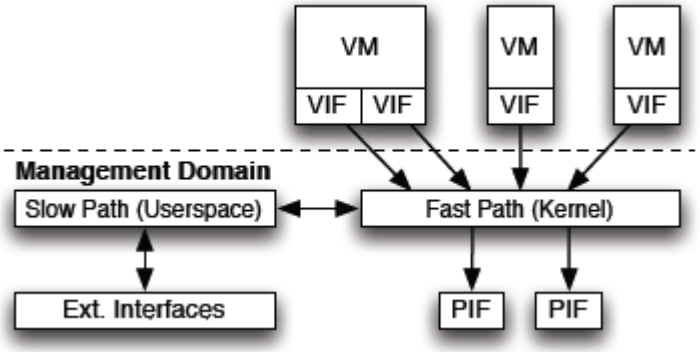
Το τυπικό μοντέλο διαδικτύωσης στο οποίο βασίζει την λειτουργία του ένα virtualized περιβάλλον βασισμένο στο Open vSwitch συνίστανται από L2 switches και IP routers. Το virtual switch επιτυγχάνει διασύνδεση μεταξύ των συνεγκατεστημένων virtual machines καθώς επίσης και με την φυσική διεπαφή. Επίσης διαθέτει μια ευέλικτη forwarding engine βασισμένη σε πίνακα που μπορεί να χρησιμοποιηθεί για λογική κατάτμηση των forwarding plane. Είναι λογισμικό ανοικτού κώδικα και είναι συμβατό με linux-based virtualization περιβάλλοντα όπως το Xen, XenServer, KVM και QEMU. Η δημιουργία virtual δικτύων με χρήση του Open vSwitch συνίσταται στην διασύνδεση των virtual interfaces (VIFs) των virtual machines με τα virtual switches, και στην διασύνδεση αυτών με τα πραγματικά interfaces (PIFs) των κόμβων στους οποίους τρέχουν.



Η υλοποίηση του Open vSwitch συνίσταται από δύο τμήματα: ένα στον πυρήνα “fast path” και από έναν userspace “slow path”. Μέσω του fast path υλοποιείται η forwarding engine ο ρόλος της οποίας είναι η έλεγχος των πακέτων ένα προς ένα, η τροποποίηση και η προώθηση τους.

Οι περισσότερες λειτουργίες υλοποιούνται εντός του slow path που τρέχει εντός του VM management domain. Στο slow path υλοποιούνται λειτουργίες που αφορούν τις τεχνικές προώθησης και εκμάθησης όπως η γνώση των MAC διευθύνσεων και η εξισορρόπηση του φορτίου στα interfaces.

Επιλέχθηκε το fast path να είναι απλό ως προς την υλοποίηση σε αντίθεση με το slow path που ενσωματώνει περισσότερες λειτουργίες.



3.7.2 Χαρακτηριστικά και δυνατότητες του Open vSwitch

Παραμετροποίηση:

Μέσω του configuration interface που διαθέτει το vSwitch μια απομακρυσμένη διαδικασία μπορεί να κάνει ανάγνωση και εγγραφή καταχωρήσεων καθώς επίσης και λήψη πληροφοριών για τις αλλαγές που λαμβάνουν χώρα όσο αφόρα την παραμετροποίηση

Forwarding Path:

Εκτός από interfaces για διαχείριση που διαθέτουν και τα φυσικά switches το Open vSwitch διαθέτει μια μέθοδο για την απομακρυσμένη διαχείριση της λειτουργίας προώθησης (forwarding path) των virtual switches. Η συγκεκριμένη μέθοδος επιτρέπει σε εξωτερικές διαδικασίες να κάνουν εγγραφές στον πίνακα προώθησης που αφορούν τον τρόπο χειρισμού των πακέτων από το switch ανάλογα με τις L2, L3, L4 ετικέτες που διαθέτουν. Το forwarding path interface βασίζεται στο πρωτόκολλο OpenFlow

Connectivity management:

Το Open vSwitch διαθέτει τοπικό interface για διαχείριση της εικονικής δικτυακής τοπολογίας. Μέσω του συγκεκριμένου interface μπορούν να γίνουν διάφορες ενέργειες που αφορούν την δημιουργία virtual switches και την διαχείριση των VIF και PIF interfaces. Έτσι κάθε εικονικό switch διαχειρίζεται ξεχωριστά από τα υπόλοιπα μέσω του interface διαχείρισης που το κάθε ένα διαθέτει. Κάθε switch διαθέτει τον δικό πίνακα προώθησης που μπορεί να παραμετροποιηθεί σε περιπτώσεις migration μιας virtual machine από έναν κόμβο του φυσικού δικτύου σε έναν άλλο.

Κινητικότητα μεταξύ των υποδικτύων:

Ένας σημαντικός περιορισμός που τίθενται στα virtual δίκτυα αφορά την δυνατότητα για migration των virtual machines από έναν κόμβο σε έναν άλλο εντός του virtualized περιβάλλοντος. Το migration μπορεί να γίνει μόνο εάν ο νέος κόμβος στον οποίο θα μετακινηθεί το virtual machine ανήκει στο ίδιο υποδίκτυο με τον προηγούμενο, θέτοντας έτσι σε σημαντικό βαθμό περιορισμούς στην επέκταση του δικτύου. Αντίθετα, το Open vSwitch ενσωματώνει δυνατότητα migration μιας virtual machine σε άλλον κόμβο που μπορεί να ανήκει σε διαφορετικό υποδίκτυο. Αυτή η κινητικότητα των virtual machines μεταξύ διαφορετικών υποδικτύων μπορεί να υποστηριχθεί με διάφορους τεχνικές. Σύμφωνα με μία από αυτές τις τεχνικές που παραπέμπει στο Mobile IP το OpenVswitch λαμβάνει όλα τα πακέτα που προορίζονται για μια virtual machine και στη συνέχεια μέσω tunneling τα προωθεί στην virtual machine. Απαιτείται παρακολούθηση των tunnels και των νέων θέσεων των virtual machines υπό την global management process.

Virtual Private Networks:

Το Open vSwitch υποστηρίζει την λειτουργία virtual private networks ως αποτέλεσμα της διασύνδεσης virtual machines πάνω από ένα ιδιωτικό virtual network που έχει υλοποιηθεί πάνω από μια μοιραζόμενη φυσική δικτυακή υποδομή. Όταν τα virtual machines βρίσκονται στον ίδιο φυσικό κόμβο ή σε ξεχωριστούς κόμβους κάθε ένας εκ των οποίων διασυνδέεται μέσω μιας κάρτας δικτύου με ένα φυσικό switch δεν χρειάζεται κάποια πρόσθετη υποστήριξη. Αντίθετα, όταν οι virtual machines είναι εγκατεστημένες σε διάφορους κόμβους και διασυνδέονται με περισσότερα τους ενός switches υπεισέρχεται μεγάλη πολυπλοκότητα στην υλοποίηση του VPN. Το Open vSwitch υποστηρίζει VLANs και GRE tunnels. Το VLAN χρησιμοποιείται για την υλοποίηση μικρής κλίμακας δικτύων ενώ τα GRE tunnels για μεγαλύτερης κλίμακας δίκτυα.

Για το Open vSwitch τα GRE tunnels χρησιμοποιούνται ως μια μέθοδος για ενθυλάκωση των πλαισίων Ethernet σε IP πακέτα ώστε να δρομολογηθούν στη συνέχεια μεταξύ των διάφορων υποδικτύων μέσω δρομολογητών. Δεδομένου ότι η global management process έχει καταχωρίσεις σχετικά με την θέση των virtual machines σε κάθε VPN, η global management process επιλέγει την καλύτερη διαδρομή για την προώθηση των πακέτων από ένα VM σε ένα άλλο με κατάλληλη τροποποίηση των flow tables στα Open vSwitches.
Συνοψίζοντας, διακρίνονται τρεις περιπτώσεις επικοινωνίας μεταξύ VMs εντός VPNs και παρατίθενται οι αντίστοιχες τεχνικές που χρησιμοποιούνται:

- Όταν οι VMs βρίσκονται στον ίδιο φυσικό κόμβο το VPN υλοποιείται εντός ενός μόνο Open vSwitch
- Όταν VMs βρίσκονται σε διαφορετικούς φυσικούς κόμβους και στο ίδιο υποδίκτυο τότε γίνεται χρήση VLANs από την management process
- Όταν VMs βρίσκονται σε διαφορετικούς φυσικούς κόμβους και σε διαφορετικά υποδίκτυα γίνεται χρήση GRE tunnels που επιτρέπουν την μεταφορά πλαισίων Ethernet μέσω IP πακέτων ώστε αυτά στη συνέχεια να μπορούν να δρομολογηθούν μεταξύ των διαφορετικών υποδικτύων.

4. Πειραματικές υποδομές

Στο κεφάλαιο αυτό θα γίνει αναφορά στις διάφορες πειραματικές πλατφόρμες που έχουν δημιουργηθεί με σκοπό την αποτίμηση και εξαγωγή χρήσιμων συμπερασμάτων σε νέες τεχνολογίες και αρχιτεκτονικές στις οποίες θα βασιστεί η ανάπτυξη του μελλοντικού διαδικτύου (Future Internet)

4.1 Violin

Το Violin [6]είναι μια virtualization αρχιτεκτονική σε επίπεδο εφαρμογής όπου απομονωμένα δίκτυα υλοποιούνται σε μορφή software πάνω σε υφιστάμενα overlay networks όπως πχ το Planetlab. Η αρχιτεκτονική του Violin περιλαμβάνει virtual end-hosts, virtual routers, virtual switches που υλοποιούνται ως λογισμικό και τρέχουν πάνω σε φυσικούς κόμβους ως virtual machines.

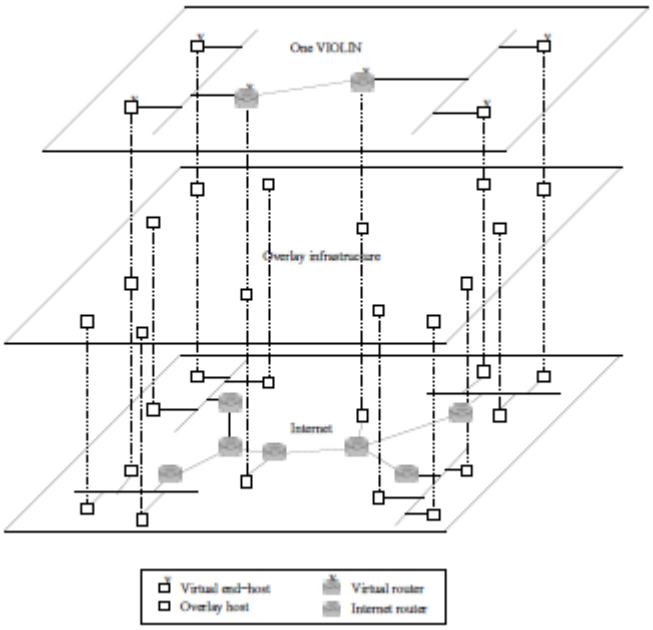
4.1.1 Χαρακτηριστικά

Τα βασικά χαρακτηριστικά [7] του Violin είναι:

- Κάθε violin έχει το δικό του χώρο διευθύνσεων IP. Κάθε κίνηση δεδομένων γίνεται αποκλειστικά εντός του violin.
- Κάθε οντότητα της αρχιτεκτονικής του Violin όπως αναφέρθηκε παραπάνω είναι υλοποιημένη σε λογισμικό εντός εικονικοποιημένου περιβάλλοντος γεγονός που προσδίδει σε ένα Violin μεγάλη ευελιξία δεδομένου ότι προσφέρεται προσθήκη, διαγραφή, migration και παραμετροποίηση κάθε οντότητας εντός του overlay δικτύου στο οποίο υλοποιείται το Violin.
- Υπηρεσίες προστιθέμενης αξίας όπως anycast, multicast, reliable multicast και active networking που δεν χρησιμοποιούνται ευρέως στο πραγματικό διαδίκτυο μπορούν να παρέχονται από ένα Violin.
- Υψηλό επίπεδο δικτυακής ασφάλειας (security) γεγονός που επιτρέπει την εκτέλεση εφαρμογών που χρήζουν υψηλών επιπέδων ασφάλειας όπως είναι οι legacy applications

4.1.2 Αρχιτεκτονική του Violin

Η αρχιτεκτονική του Violin παρουσιάζεται στο παρακάτω σχήμα



Το χαμηλότερο επίπεδο απεικονίζει το πραγματικό IP δίκτυο, το μεσαίο επίπεδο το overlay δίκτυο και το υψηλότερο επίπεδο ένα Violin δίκτυο. Όλες οι οντότητες του Violin είναι υλοποιημένες σε λογισμικό και τρέχουν σε overlay hosts. Ένα Violin δίκτυο αποτελείται από end-hosts, LANs και routers

- Ένας virtual end-host (vHost) είναι μια virtual machine που τρέχει σε έναν πραγματικό overlay host. Σε έναν overlay host μπορούν να τρέχουν και άλλοι vHosts που ανήκουν σε διαφορετικά Violins.
- Ένα virtual LAN (vLAN) προκύπτει από τη σύνδεση ενός virtual switch (vSwitch) με πολλούς vHosts μέσω UDP tunnels [6]
- Ένας virtual router (vRouter) είναι μία virtual machine με πολλαπλά virtual interfaces (vNICs). Ένα vRouter διασυνδέει δύο ή περισσότερα vLANs.

4.1.3 Εφαρμογές του Violin

Το Violin όντας μια καινοτόμος δικτυακή αρχιτεκτονική μπορεί να χρησιμοποιηθεί:

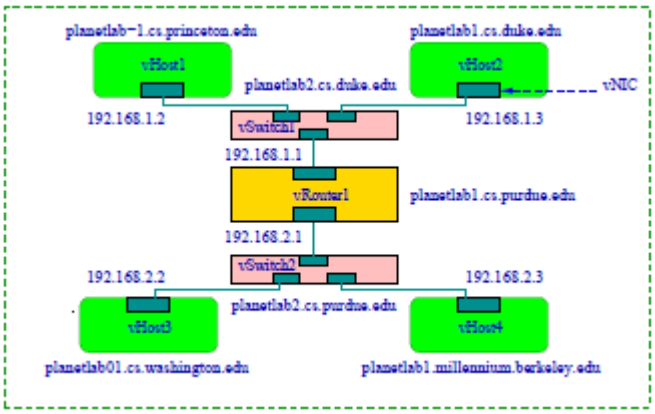
1. ως testbed σε πειράματα που αφορούν το επίπεδο δικτύου. Ως testbed προσφέρει ρεαλιστικές συνθήκες διότι διαθέτει πολλά στοιχεία ενός πραγματικού δικτύου όπως switches, routers κλπ.

2. για τη δημιουργία δικτύου προσανατολισμένου στη παροχή προηγμένων δικτυακών υπηρεσιών όπως IP multicast και anycast που θα δώσουν ώθηση σε εφαρμογές όπως το video conferencing.

4.1.4 Πλεονεκτήματα χρήσης Violin

Απομόνωση:

Το Violin είναι πλήρως απομονωμένο από το φυσικό IP δίκτυο όπως ακριβώς μια virtual machine και ο host στον οποίο τρέχει. Κάθε οντότητα του Violin είναι πλήρως απομονωμένη από το internet. Για την περιγραφή της ιδιότητας αυτής θα χρησιμοποιήσουμε την παρακάτω τοπολογία δικτύου που απεικονίζει ένα Violin που έχει υλοποιηθεί στο PlanetLab και αποτελείται από δύο vLans.



Το πρώτο Vlan αποτελείται από το vSwitch1, vHost1 και vHost2 ενώ το δεύτερο από το vSwitch2, vHost3 και vHost4. Αν στον κόμβο vHost1 εκτελέσουμε την εντολή traceroute αναφερόμενοι στον κόμβο vHost3 τότε ως επόμενο hop θα έχουμε το vRouter1 παρόλο υφίστανται και άλλα routers στην υποδομή του Planetlab. Η δικτυακή απομόνωση (network isolation) συνεπάγεται για τον διαχειριστή του δικτύου αποκλειστικά πρόσβαση και διαχείριση του συγκεκριμένου Violin και μόνο αυτού. Επίσης οι IP διευθύνσεις εντός ενός Violin μπορούν να συμπίπτουν με IP διευθύνσεις του δικτύου υποδομής και μπορούν να διαφέρουν και στην έκδοση τους (IPv4,IPv6). Επιπρόσθετα, μια δικτυακή επίθεση εναντίον ενός Violin δεν θα επηρεάσει το υπόλοιπο Internet. Επιπλέον, αν οι κόμβοι του δικτύου υποδομής παρέχουν υποστήριξη για ποιότητα υπηρεσίας, το Violin είναι σε θέση να εξασφαλίσει απομόνωση των πόρων (resource isolation) για τοπικούς πόρους όπως η CPU,

η μνήμη κλπ των Violin οντοτήτων καθώς και απομόνωση για το εύρος ζώνης που διατίθεται για την σύνδεση των οντοτήτων μεταξύ τους. Άλλο ένα χαρακτηριστικό του violin συνδέεται με την system-level virtualization και την απομόνωση που συνεπάγεται αφορά την δυνατότητα εκτέλεσης κατανεμημένων επισφαλών εφαρμογών και γενικότερα επισφαλών δικτυακών πειραμάτων. Ακόμα όμως και εφαρμογές που απαιτούν ιδιαίτερη ασφάλεια μπορούν να τρέξουν στο Violin ώστε να αποφύγουν την διαρροή πληροφοριών καθώς και τυχόν εξωτερικές δικτυακές επιθέσεις.

Παροχή δικτυακών υπηρεσιών:

Το Violin αποτελεί ένα σημαντικό project για την παροχή προηγμένων δικτυακών υπηρεσιών. Υπηρεσίες όπως το IP multicast, IP anycast και το Active Networking που δεν παρέχονται ακόμα από πολλά πρωτόκολλα του διαδικτύου μπορούν να βρουν στο Violin το ιδανικό περιβάλλον για να αναπτυχθούν.

Εύκολη παραμετροποίηση:

Το Violin επιτρέπει εύκολη παραμετροποίηση των οντοτήτων του. Σε αντίθεση με ένα παραδοσιακό φυσικό δίκτυο IP, το Violin αποτελείται από οντότητες που έχουν υλοποιηθεί αποκλειστικά σε επίπεδο λογισμικού όντας virtual machines καθιστώντας έτσι εύκολη την παραμετροποίηση τους. Έτσι τα vRouters, vSwitches, vHosts της virtual τοπολογίας μπορούν να προστίθεται, να αφαιρούνται ή να αλλάζουν θέση στο φυσικό δίκτυο (migration). Η ευελιξία που προσφέρει το Violin όσο αφορά τα virtual interfaces καθίσταται ιδιαίτερα χρήσιμη σε περιπτώσεις κατανεμημένων εφαρμογών όπου αυτά τα interfaces δημιουργούνται ή αφαιρούνται ανάλογα με τη ζήτηση από τις εφαρμογές που τρέχουν πάνω σε αυτό. Το γεγονός αυτό βοηθά σε μεγάλο βαθμό στην κλιμάκωση ενός Violin καθώς μπορεί να επεκτείνεται με δυναμικό τρόπο και on-demand.

4.1.5 Υλοποίηση των οντοτήτων του Violin

Virtual Machine:

Κάθε οντότητα του Violin (virtual end-host, vSwitch, vRouter) είναι υλοποιημένη ως virtual machine. Για την υλοποίηση των virtual machine έγινε χρήση UML (User Mode Linux) ως τεχνολογίας δημιουργίας virtual machine. Η συγκεκριμένη τεχνολογία επιτρέπει στις περισσότερες εφαρμογές που τρέχουν σε Linux περιβάλλον να τρέχουν σε virtual machines

χωρίς τροποποίηση άρα στην περίπτωση του Violin συναντάμε full virtualization. Λόγω του ότι η τεχνολογία UML επιβάλλει οι virtual διεπαφές καθώς και οι virtual ζεύξεις να βρίσκονται εντός ενός φυσικού host κρίθηκε απαραίτητη η επέκταση του UML μέσω διασυνδέσεων φυσικών hosts στο επίπεδο μεταφοράς με χρήση του UDP πρωτοκόλλου. Για παράδειγμα για την διασύνδεση ενός virtual host με ένα vSwitch το guest OS για τον vHost

Virtual Switch:

Ένα vSwitch δημιουργείται για τον ορισμό ένα vLAN και ο ρόλος του είναι η προώθηση πακέτων εντός του εικονικού στρώματος ζεύξης δεδομένων. Οι πόρτες του vSwitch μπορούν να δημιουργούνται με δυναμικό τρόπο έπειτα από αίτημα ενός νέου vHost για σύνδεση. Τέτοιου είδους αιτήμα καταφθάνουν σε μια συγκεκριμένη πόρτα του vSwitch που ονομάζεται Control Port.

4.2 VINI

Το VINI είναι μια εικονική δικτυακή υποδομή που επιτρέπει στους ερευνητές δικτύων την αξιολόγηση πρωτοκόλλων και υπηρεσιών με αρκετά ικανοποιητικό τρόπο με δυνατότητα μεταβολής πολλών παραμέτρων που αφορούν το δίκτυο (φορτίο του δικτύου, διάφορα δικτυακά συμβάντα, όπως η κατάρρευση μιας ζεύξης). Μέσω του VINI καθίσταται αποτελεσματική η αξιολόγηση κάθε νέας προτεινόμενης τεχνολογίας αφού μπορεί να εκπονηθεί μια σειρά από πειραματικές μετρήσεις πάνω σε κάθε δυνατή τοπολογία εικονικού δικτύου που μπορεί να υπάρξει με δεδομένο ένα πραγματικό δίκτυο. Το VINI όπως και άλλα testbeds έρχεται ως απάντηση στις επιφυλάξεις των διαχειριστών δικτύων να υιοθετήσουν νέες τεχνολογίες στα δίκτυα τους φοβουμένοι την χειροτέρευση της ποιότητας υπηρεσίας (Qos) που αυτά προσφέρουν.

4.2.1 Προδιαγραφές VINI

Η σχεδίαση του VINI έγινε έτσι ώστε να ικανοποιεί κάποιες προδιαγραφές σε σχέση με άλλα υπάρχοντα περιβάλλοντα εξομοίωσης:

- Να επιτρέπει τη χρήση των υπαρχόντων λογισμικών δρομολόγησης
- Να επιτρέπει την χρήση δικτυακών συμβάντων όπως η διακοπή μιας σύνδεσης μεταξύ δύο κόμβων

- Να υποστηρίζει τη χρήση πραγματικής δικτυακής κίνησης μεταξύ των διάφορων τερματικών στοιχείων του δικτύου, ώστε να μπορέσει να γίνει μέτρηση παραμέτρων που αφορούν την επίδοση του δικτύου
- Να εξομοιώνει με ρεαλιστικό τρόπο τις δικτυακές συνθήκες

Η εικονική δικτυακή υποδομή υλοποιείται με χρήση του National Lambda Rail (NLR) και του Abilene Internet2 δικτύων κορμού. Επίσης θα διαθέτει IP διευθύνσεις ορατές από οποιαδήποτε άλλο δίκτυο. Για την υλοποίηση του VINI χρησιμοποιήθηκαν το XORP για δρομολόγηση, το Click για προώθηση πακέτων και NAT (Network Address Translation), Open VPNs servers για την σύνδεση με τελικούς χρήστες και το rcc.
Ο έλεγχος αναφέρεται στην δυνατότητα ενσωμάτωσης στα πειράματα εξωγενών παραγόντων που αφορούν την λειτουργία του δικτύου (όπως πτώση μιας ζεύξης, μεταβολές στη κίνηση του δικτύου). Κάτι τέτοιο είναι πολύ σημαντικό δεδομένου ότι πολλές φορές οι ερευνητές κατά την υλοποίηση μιας νέας τεχνολογίας (πρωτοκόλλου ή αρχιτεκτονικής) θέλουν να δουν πως συμπεριφέρεται ανάλογα με τις διάφορες μεταβολές που γίνονται στο δίκτυο. Το VINI παρέχει δυνατότητες ελέγχου ανάλογες προσομοιωτών (simulators) όπως του ns-2, SSFNet και εξομοιωτών (emulation testbeds)
Ο ρεαλισμός αναφέρεται στην ικανότητα των ερευνητών δικτύων να αξιολογούν ένα πρωτόκολλο ή μια αρχιτεκτονική κάτω από πραγματικές συνθήκες που συναντώνται σε ένα πραγματικό δίκτυο όπως (μεταβολές στην τοπολογία, αστοχίες και μεταβαλλόμενη κίνηση του δικτύου) . Με αυτό τον τρόπο μπορεί να γίνει επακριβής αποτίμηση των υπό σχεδίαση πρωτοκόλλων ή αρχιτεκτονικών. Το VINI επιτρέπει στους ερευνητές να αποτιμήσουν τις υλοποιήσεις τους σε πραγματικά δίκτυα.
Το VINI είναι κατάλληλο για υλοποίηση πειραμάτων που απαιτούν και ρεαλισμό και έλεγχο. Τα πειράματα αυτά χωρίζονται σε δύο κατηγορίες: τα ελεγχόμενα πειράματα και τα long-running deployment studies

Κατά την υλοποίηση του VINI ελήφθησαν υπόψη ορισμένοι βασικοί παράγοντες:
1)Θα πρέπει να υποστηρίζεται σύνδεση σημείο προς σημείο μεταξύ οποιοδήποτε κόμβων του δικτύου υποδομής ώστε ο ερευνητής να έχει στη διάθεση του οποιαδήποτε δυνατή τοπολογία. Για κάθε πείραμα θα πρέπει να χρησιμοποιούνται μοναδικές εικονικές διεπαφές (interfaces) που θα είναι λιγότερες από τις φυσικές διεπαφές.

2)Το δίκτυο υποδομής πρέπει να επιτρέπει την εκτέλεση λογισμικού δρομολόγησης στο εικονικό δίκτυο. Στο σημείο αυτό υπάρχει δυσκολία λόγω του ότι κάθε εικονικός κόμβος διαφέρει σε λειτουργίες από τον κόμβο υποδομής

3)Κάθε εικονικό δίκτυο με δεδομένους πίνακες δρομολόγησης και προώθησης θα πρέπει να μπορεί να τα ανταλλάσει κίνηση με υφιστάμενα πραγματικά δίκτυα. Επειδή μάλιστα κάθε κόμβος του δικτύου υποδομής μπορεί να χρησιμοποιείται σε πολλά πειράματα κατεπέκταση σε πολλές εικονικές τοπολογίες χρειάζεται για κάθε πείραμα να αναπτυχθεί ένας εικονικός κόμβος (virtual machine) με το δικό του πίνακα δρομολόγησης και προώθησης. Οι διαδικασίες δρομολόγησης πρέπει να αφορούν και προορισμούς εκτός του δίκτυου. Οι πίνακες προώθησης θα υλοποιηθούν ως οντότητες του Click και οι πίνακες δρομολόγησης ως οντότητες του XORP τεχνολογίες που θα επεξηγηθούν στη συνέχεια.

4)Θα πρέπει να μπορούν να εργαστούν πολλοί ερευνητές πάνω στο ίδιο εικονικό δίκτυο δεδομένου ότι ικανοποιούνται οι παραπάνω 3 προδιαγραφές. Δεν θα πρέπει η εκτέλεση του ενός πειράματος να επηρεάζει την εκτέλεση ενός άλλου.

5)Χρειάζεται πλήρης απομόνωση των πόρων μεταξύ των διάφορων εικονικών κόμβων που τρέχουν (VMs) που τρέχουν εντός ενός πραγματικού κόμβου ώστε κάθε πείραμα να μην επηρεάζει την εκτέλεση άλλου πειράματος. Κάθε εικονικός κόμβος επιπλέον χρειάζεται την δική του IP διεύθυνση και name space

4.2.2 Βασικά χαρακτηριστικά του VINI

Ως φυσική υποδομή για την υλοποίηση του VINI επιλέχθηκε το PlanetLab λόγω των εκτεταμένων δικτυακών υποδομών που διαθέτει αλλά και την υποστήριξης τεχνικών virtualization. Το Planetlab απομονώνει τα πειράματα σε VServers που είναι ένα τμήμα ενός server με το δικό του namespace γεγονός που επιτρέπει την ταυτόχρονη εκτέλεση πολλών πειραμάτων. Κατά συνέπεια μπορούν να τρέχουν πολλαπλά pl-vini ταυτόχρονα σε ένα Planetlab σε διαφορετικά slices. Η δικτυακή απομόνωση στο PlanetLab διασφαλίζεται από το VNET που πολυπλέκει και αποπολυπλέκει την εισερχόμενη και εξερχόμενη δικτυακή κίνηση σε κάθε κόμβου του PlanetLab. Μια βασική καινοτομία του VINI είναι η υλοποίηση πειραμάτων με χρήση δεδομένων που προέρχονται από εξωτερικούς κόμβους ή και εφαρμογές που τρέχουν στο ίδιο φυσικό κόμβο. Για την μεταφορά των δεδομένων αυτών στο VINI χρησιμοποιείται το OpenVPN ως μηχανισμός διοχέτευσης των δεδομένων στο virtualized περιβάλλον του VINI. Κάθε VINI διαθέτει ένα πλήθος από κόμβους εισόδου

(ingress nodes) στους οποίους με την βοήθεια του OpenVPN γίνεται μεταφορά δεδομένων από την φυσική δικτυακή υποδομή στην virtualized. Πιο συγκεκριμένα σε κάθε κόμβο εγκαθίσταται ένας OpenVPN client

4.2.3 Click

Το Click [22] είναι μια αρχιτεκτονική λογισμικού για την υλοποίηση ευέλικτων και παραμετροποιήσιμων δρομολογητών. Ένας δρομολογητής Click αποτελείται από elements που είναι στοιχεία επεξεργασίας πακέτων. Τα elements υλοποιούν απλές λειτουργίες όπως ταξινόμηση πακέτων, τοποθέτησή τους στην ουρά, χρονοπρογραμματισμός και δημιουργία διεπαφών με τις δικτυακές συσκευές. Διάφορα χαρακτηριστικά δίνουν στα elements πιο πολύπλοκες μορφές, αλλά με περισσότερες δυνατότητες, όπως για παράδειγμα τα pull connections που μοντελοποιούν τη ροή των πακέτων που μεταδίδονται από τις δικτυακές συσκευές ή το flow-based router context, το οποίο επιτρέπει σε ένα element να εντοπίσει άλλα. Με αυτό τον τρόπο μπορούν να εκτελεστούν πιο σύνθετες λειτουργίες, με λογική παρόμοια με αυτή του UNIX όπου πιο σύνθετες εφαρμογές αναπτύσσονται απευθείας από πιο απλές χρησιμοποιώντας pipes.

Αρχιτεκτονική Click:
Το element αντιπροσωπεύει μία μονάδα επεξεργασίας του δρομολογητή. Πρακτικά είναι περισσότερο ένας απλός, στοιχειώδης υπολογισμός, π.χ. μείωση του πεδίου TTL ενός IP πακέτου, παρά πιο μεγάλος και πολύπλοκος, όπως η δρομολόγησή του. Η παραμετροποίηση ενός δρομολογητή Click είναι ένας κατευθυνόμενος γράφος όπου τα elements είναι οι κορυφές του. Μια ακμή, δηλαδή ένα connection, μεταξύ δύο elements αντιπροσωπεύει ένα πιθανό μονοπάτι στη μετάδοση του πακέτου. Οποιαδήποτε ενέργεια γίνεται από το λογισμικό του δρομολογητή, για παράδειγμα αναζήτηση σε πίνακα δρομολόγησης ή αρίθμηση πακέτων, ενθυλακώνεται σε ένα element. Ο χρήστης καθορίζει τι θα κάνει ο δρομολογητής Click επιλέγοντας τα elements τα οποία θα χρησιμοποιηθούν καθώς και τις συνδέσεις μεταξύ τους. Από τεχνικής πλευράς ένα element μέσα στον δρομολογητή είναι ένα αντικείμενο σε C++. Οι συνδέσεις υλοποιούνται με δείκτες στα αντικείμενα των elements και η μετάδοση ενός πακέτου μέσω μιας σύνδεσης είναι μια απλή κλήση εικονικής συνάρτησης.
Οι ιδιότητες ενός element που είναι άξιες αναφοράς είναι:

- **Element class:** κάθε element ανήκει σε μια element κλάση. Αυτό προσδιορίζει τον κώδικα που θα εκτελεστεί όταν ένα element επεξεργάζεται ένα πακέτο, καθώς και την αρχικοποίησή αυτού και των δεδομένων του.
- **Θύρες:** ένα element μπορεί να έχει ένα πλήθος από εισερχόμενες και εξερχόμενες θύρες. Κάθε σύνδεση γίνεται μεταξύ της εξερχόμενης θύρας ενός element και της εισερχόμενης θύρας ενός άλλου. Επίσης είναι δυνατόν να υπάρχουν δευτερεύουσες εξερχόμενες θύρες για να απορρίπτουν εσφαλμένα πακέτα.
- **Συμβολοσειρά παραμετροποίησης:** Η προαιρετική συμβολοσειρά παραμετροποίησης περιέχει επιπρόσθετα γνωρίσματα που ανατίθενται στο element κατά την εκκίονηση του δρομολογητή. Αρκετές element κλάσεις χρησιμποποιούν αυτά τα γνωρίσματα για να θέσουν την κατάσταση για κάθε element και να ρυθμίσουν τη συμπεριφορά τους.
- **Διεπαφές μεθόδων:** Καθε element υποστηρίζει μία ή περισσότερες διεπαφές μεθόδων. Όλα τα elements υποστηρίζουν τη διεπαφή μεταφοράς πακέτου αλλά κάποια elements δημιουργούν αυθαίρετα τις δικές του διεπαφές, για παράδειγμα μια ουρά μπορεί να δημιουργήσει μια διεπαφή που ανακοινώνει το μήκος της ουράς. Τα elements επικοινωνούν μέσω των διεπαφών, οι οποίες μπορεί να περιέχουν είτε δεδομένα έιτε μεθόδους.

Συνδέσεις Push και Pull:

Το Click υποστηρίζει δύο είδη συνδέσεων, τις push και pull. Σε μια push σύνδεση, τα πακέτα ξεκινούν από το element πηγή και καταλήγουν στο element προορισμό. Αντίθετα, σε μια pull σύνδεση, το element προορισμού εκκινεί τη μετάδοση των πακέτων. Ζητά από την πηγή να επιστρέψει ένα πακέτο ή ένα δείκτη null αν κανένα πακέτο δεν είναι διαθέσιμο. Και οι δύο αυτές μορφές μετάδοσης πακέτων υλοποιούνται από μια κλήση εικονικής συνάρτησης.

Ο τύπος επεξεργασίας μιας σύνδεσης καθορίζεται από τις θύρες στα τερματικά σημεία. Κάθε θύρα σε ενα δρομολογητή είναι είτε push είτε pull και οι συνδέσεις είναι μεταξύ δύο θυρών push ή δύο θυρών pull. Δεν επιτρέπονται συνδέσεις μεταξύ θυρών push και pull. Κατά την εκκίνηση του δρομολογητή τα elements καθορίζουν τις θύρες τους. Μπορούν επίσης να δημιουργήσουν agnostic θύρες, οι οποίες συμπεριφόρονται ως push ή pull όταν συνδέονται αντίστοιχα με θύρες push και pull.

Η επεξεργασία push λαμβάνει χώρα όταν τα πακέτα φθάνουν στον δρομολογητή και αυτός θα πρέπει να τα διαχειριστεί κατάλληλα καθ' όσο αυτά φθάνουν. Η επεξεργασία pull λαμβάνει χώρα όταν ο δρομολογητής πρέπει να ελέγξει το χρόνο της επεξεργασίας

πακέτων. Για παράδειγμα, ο δρομολογητής μπορεί να μεταδίδει μόνο όταν η συσκευή μετάδοσης ειναι έτοιμη. Στο Click οι συσκευές μετάδοσης είναι elements με ένα εισερχόμενο pull, άρα αυτά εκεινούν τη μετάδοση, και μπορούν να ζητούν πακέτα μόνο όταν αυτές ειναι έτοιμες. Επίσης η επεξεργασία pull μοντελοποιεί την απόφαση χρονοπρογραμματισμού, επιλέγοντας πιο θα είναι το επόμενο πακέτο προς αποστολή.

Αποθήκευση πακέτων:

Τα Click elements δεν έχουν ουρές αποκλειστικά για τις εισερχόμενες και εξερχόμενες θύρες. Οι ουρές υλοποιούνται σαν ξεχωριστά αντικείμενα. Αυτό δίνει ευελιξία στον σχεδιαστή του δρομολογητή γιατί μπορεί να καθορίσει αποκλειστικά τον τρόπο με τον οποίο αποθηκεύονται τα πακέτα. Επίσης, επιτρέπει δύσκολες παραμετροποιήσεις, όπως για παράδειγμα μια ουρά που τροφοδοτεί πολλές συσκευές ταυτόχρονα ή ένας διαμορφωτής κίνησης που παρεμβάλλεται στη διαδρομή προς μια συσκευή. Οι ουρές απαιτούν και τις δύο συνδέσεις push κια pull. Η εισερχόμενη push σύνδεση τοποθετεί τα πακέτα στην ουρά, ενώ η εξερχόμενη pull τα αφαιρεί από την ουρα.

Χρονοπρογραμματισμός CPU:

Το Click χρονοπρογραμματίζει το CPU του δρομολογητή με μια ουρά εργασιών, διεκπαιραιώνοντας μια εργασία τη φορά. Κάθε εργασία είναι ένα element, άρα το element είναι ταυτόχρονα μονάδα χρονοπρογραμματισμού και επεξεργασίας πακέτων. Μια εργασία μπορεί να εκκινήσει μια αυθαίρετη ακολουθία από αιτήσεις push και pull. Άλλη μια δομή χειρίζεται τα χρονικά γεγονότα. Κάθε element μπορεί να έχει οποιοδήποτε αριθμό από ενεργούς χρονομετρητές, όπου κάθε χρονομετρητής καλεί μια αυθαίρετη μέθοδο όταν εκκινεί.

Το Click τρέχει σε ένα νήμα. Έτσι οποιαδήποτε μέθοδος μεταφοράς πακέτου (push ή pull) πρέπει να επιστρέψει πριν αρχίσει κάποια άλλη εργασία. Ο δρομολογητής θα συνεχίσει να επεξεργάζεται κάθε πακέτο από element σε element κατά μήκος ενός μονοπατιού στο γράφο του δρομολογητή, έως ότου αυτό αποθηκευθεί ή απορριφθεί.

Flow-Based Router Context:

Αν ένα element A θέλει να χρησιμοποιήσει μια διεπαφή μεθόδου ενός άλλου element B πρώτα θα πρέπει να εντοπίσει το B. Οι συνδέσεις λύνουν αυτό το προβλημα στη μεταφορά πακέτου. Διαφορετικά θα πρέπει να χρησιμοποιηθεί ένας αυτόματος μηχανισμός, το flow-based router context. Το Flow-based router context περιγράφει το που θα καταλήξουν τα

πακέτα που ξεκινούν από ένα δεδομένο element, μετά από κάποιο αριθμό μεταφορών, ή αντίστροφα από που προήλθαν τα πακέτα που αφίχθησαν σε δεδομένο element. Το γεγονός αυτό γενικεύει τις συνδέσεις, οι οποίες περιγράφουν ακριβώς μια μεταφορά.συνεπώς το σύστημα αποκτά μια ευρωστία, αφού προσαρμόζεται αυτόματα σε αλλαγές των ρυθμίσεων του δρομολογητή. Το flow-based router context υπολογίζεται από ένα απλό αλγόριθμο ροής δεδομένων. Τα elements γενικά ζητούν το flow-based router context μια φορά και κατά την εκκίνηση του δρομολογητή και αποθηκεύουν τα αποτελέσματά του για μελλοντική γρήγορη αναφορά.

Υλοποίηση των elements:

Κάθε Click element κλάση αντιστοιχεί σε μια υποκλάση της κλάσης Element της C++, η οποία έχει περίπου είκοσι εικονικές συναρτήσεις. Μόνο τρεις εικονικές συναρτήσεις χρησιμοποιούνται κατά τη διάρκεια λειτουργίας του δρομολογητή, η push, η pull και η run_scheduled.

Σχεδιασμός των elements και αρχιτεκτονικοί περιορισμοί:

Γενικά είναι προτιμότερα μικρά και απλά elements παρά elements με πολύπλοκες προδιαγραφές. Παρόλα αυτά τα μικρά elements δεν είναι κατάλληλα για παντός είδους προβλήματα, μιας και η βασική αρχή του Click είναι η ροή πακέτων. Πολύπλοκα elements χρειάζονται όταν ο έλεγχος ή η ροή δεδομένων δεν συνάδει με τη ροή πακέτων. Για παράδειγμα, επεξεργασία πολύπλοκου πρωτοκόλλου συχνά απαιτεί ένα πολύπλοκο element. Ένα πρωτόκολλο δρομολόγησης οπως το BGP δεν μπορεί να διασπαστεί σε πολλές απλές διαδικασίες μεταξύ των οποίων να ρέουν πακέτα. Ένας παραδοσιακός δρομολογητής περιέχει διαμοιραζόμενες δομές που δεν εμπλέκονται στην προώθηση πακέτων, όπως πίνακες δρομολόγησης, στατιστικά δικτύου κ.α.. Στο Click, αυτές οι δομές εμπεριέχονται στο μονοπάτι της προώθησης πακέτου. Οι πίνακες δρομολόγησης IP και η προσωρινή μνήμη ARP ενθυλακώνονται στα elements που παίρνουν αποφάσεις δρομολόγησης και τα στατιστκά στοιχεία βρίσκονται τοπικά στα elements που είναι υπεύθυνα για τη συλλογή τους.

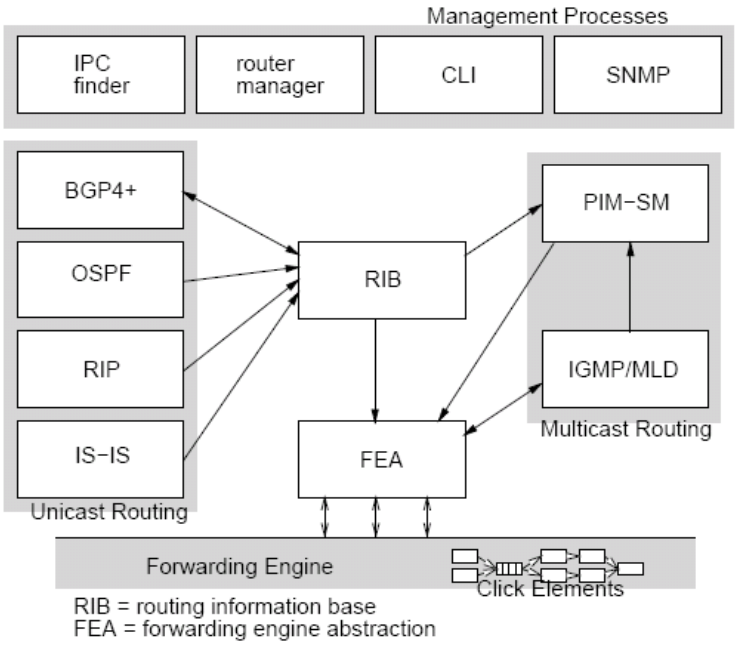
Στο Click δε μπορεί να χρονοπρογραμματίσει το CPU για μεμονωμένες ροές. Το flow-based router context είναι κατάλληλο για πολλές περιπτώσεις όπου πρέπει να εντοπιστεί μια διεπαφή μεθόδου, αλλά δεν είναι πάντα αρκετά ακριβές. Για παράδειγμα σε μια παραμετροποίηση, όπου δεν εμφανίζονται συχνά λάθος πακέτα, αυτά στέλνονται σε ΄συνολο από elements που χειρίζονται τα σφάλματα. Τα elements που στέλνουν πακέτα και

χρησιμοποιούν flow-based router context θα περιλαμβάνουν πάντα τα λανθασμένα πακέτα στην αναζήτησή τους, κάτι το οποίο δεν είναι επιθυμητό. Τα περισσότερα elements υιοθετούν μια απλή λύση κατά την οποία ο χρήστης μπορεί να παραβλέψει το flow-based router context με ένα γνώρισμα παραμετροποίησης.

4.2.4 XORP

Το XORP (eXtensible Open Router Platform) [23] χωρίζεται σε δύο υποσυστήματα. Το υποσύστημα ανώτερου επιπέδου, ονομάζεται και επίπεδο χρήστη, αποτελείται από τα πρωτόκολλα δρομολόγησης μαζί με πληροφορία δρομολόγησης και υποστηρικτικές διεργασίες. Το υποσύστημα κατώτερου επιπέδου, το οποίο αρχικά τρέχει μέσα σε ένα πυρήνα λειτουργικού συστήματος, χειρίζεται το μονοπάτι προώθησης και παρέχει APIs στα οποία έχει πρόσβαση το ανώτερο επίπεδο. Ο σκοπός είναι για όλο τον κώδικα του ανώτερου επιπέδου να είναι άγνωστος αναφορικά με τις λεπτομέρειες του μονοπατιού προώθησης. Για το επίπεδο χρήστη, υπάρχει μια αρχιτεκτονική πολλαπλών διεργασιών με μια διεργασία για κάθε πρωτόκολλο δρομολόγησης, μαζί με επιπρόσθετες διεργασίες για διαχείριση, παραμετροποίηση και συγχρονισμό. Ο μηχανισμός διαδιεργασιακής επικοινωνίας έχει σχεδιαστεί για να προσφέρει μεγαλύτερη επεκτασιμότητα. Αυτός ο μηχανισμός ονομάζεται XORP Resource Locators (XRLs) και είναι εννοιολογικά παρόμοιος με τα URLs. Ο μηχανισμός προσφέρει ανακατεύθυνση, αξιοπιστία και επεκτασιμότητα και η από φύση τους εύκολη και κατανοητή γραφή του, καθιστά την ενσωμάτωσή του σε scripting γλώσσες πολύ απλή.

Το κατώτερο επίπεδο χρησιμοποιεί το δρομολογητή Click, ένα επεκτάσιμο εργαλείο για επεξεργασία πακέτων. Μπορεί επίσης να υποστηρίξει εναλλακτικά μονοπάτια προώθησης, όπως το FreeBSD μονοπάτι προώθησης με AltQ επεκτάσεις ουρών ή ένα εναλλακτικό μονοπάτι, όπως το Scout. Κάποιες αποφάσεις αναφορικά με την επιλογή των μονοπατιών μπορεί να επηρεάσουν τη λειτουργικότητα προς τους τελικούς χρήστες..



Στο παραπάνω σχήμα φαίνεται πώς συνεργάζονται οι διεργασίες ενός XORP δρομολογητή επιπέδου χρήστη και το μονοπάτι προώθησης του Click. Οι διαμοιραζόμενες διεργασίες επιπέδου χρήστη είναι από τα πιο σημαντικά χαρακτηριστικά της αρχιτεκτονικής του XORP. Τέσσερις βασικές διεργασίες είναι αξιοσημείωτες: το router manager, το finder, το routing information base και το forwarding engine abstraction.

Η διεργασία router manager διαχειρίζεται το δρομολογητή στο σύνολό του. Διατηρεί πληροφορία παραμετροποίησης, εκκινεί άλλες διεργασίες, όπως πρωτόκολλα δρομολόγησης και επανεκκινεί διεργασίες που απέτυχαν όποτε κρίνεται απαραίτητο.

Η διεργασία finder αποθηκεύει αντιστοιχίες μεταξύ αιτήσεων από εφαρμογές, όπως για παράδειγμα το πόσες διεπαφές έχει ο συγκεκριμένος δρομολογητής, και τις κλήσεις IPC που είναι απαραίτητες για να ικανοποιηθούν αυτές οι αιτήσεις. Με λίγα λόγια όταν μια εφαρμογή θέλει να καλέσει ενα IPC συμβουλεύεται το finder για να μάθει πως θα το κάνει. Η εφαρμογή τυπικά αποθηκεύει προσωρινά αυτή την πληροφορία για ενδεχόμενη μελλοντική χρήση. Επιπλέον το finder δίνει οδηγίες στις εφαρμογές για να ενημερώσουν την πληροφορία επικοινωνίας. Έτσι είναι εύκολη η αλλαγή του τρόπου με τον οποίο οι αιτήσεις της εφαρμογής σε πραγματικό χρόνο.

Η διεργασία routing information base (RIB) λαμβάνει διαδρομές από τις διεργασίες δρομολόγησης και αυθαίρετα επιλέγει ποιες από αυτές θα πρέπει να συμπεριληφθούν στο μονοπάτι προώθησης ή να αναδιανεμηθούν μεταξύ των υπολοίπων διεργασιών δρομολόγησης.

Η διεργασία forwarding engine abstraction (FEA) διαχειρίζεται το μονοπάτι προώθησης. Το FEA αφαιρεί τις λεπτομέρειες του πως υλοποιείται το μονοπάτι του δρομολογητή και σαν αποτέλεσμα οι διεργασίες δρομολόγησης είναι αδιαφανής ως προς το αν η προώθηση γίνεται με το Click, έναν κλασσικό πυρήνα UNIX ή με μια άλλη εναλλακτική μέθοδο. Το FEA χειρίζεται τις δικτυακές διεπαφές και τον πίνακα προώθησης στο δρομολογητή και παρέχει πληροφορίες στις διεργασίες σχετικές με τις ιδιότητες των διεπαφών και συμβάντα που λαμβάνουν χώρα σε αυτές. Όπως και με το finder, οι διεργασίες XORP μπορούν να παρακάμψουν το FEA όποτε αυτό απαιτείται.

4.3 GENI

Το GENI (Global Environment for Network Innovations) [24] είναι μια ερευνητική δικτυακή υποδομή που χρηματοδοτείται από το National Science Foundation για να υποστηρίξει πειραματική έρευνα στην επιστήμη των δικτύων και τη μηχανική.
Οι κύριες έννοιες της υποδομής του GENI είναι οι εξής:

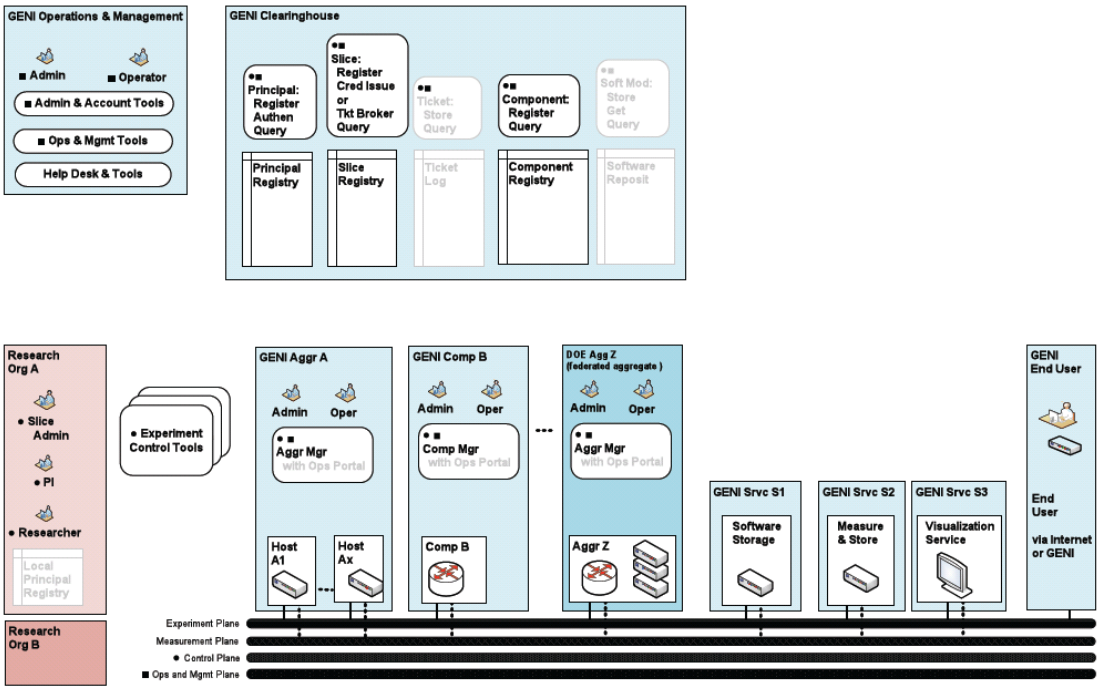
- Προγραμματισιμότητα: οι ερευνητές μπορούν να μεταφορτώσουν λογισμικό στους κόμβους του GENI ώστε να ελέγξουν τον τρόπο με τον οποίο οι κόμβοι συμπεριφέρονται
- Εικονικοποίηση και άλλες μορφές διαμοιραζόμενων πόρων: όποτε είναι εφικτό οι κόμβοι υλοποιούν εικονικές μηχανές, οι οποίες επιτρέπουν στους ερευνητές να διαμοιράζονται ταυτόχρονα την υποδομή και κάθε πείραμα τρέχει στο δικό του απομονωμένο slice
- Διαφορετικά μέρη του GENI ανήκουν ή χρησιμοποιούνται από διαφορετικούς οργανισμούς
- Πειράματα που βασίζονται σε slice: τα πειράματα GENI είναι διασυνδεδεμένα σύνολα από δεσμευμένους πόρους σε διάφορες τοποθεσίες. Οι ερευνητές δεσμεύουν, παραμετροποιούν, προγραμματίζουν, λειτουργούν, διαχειρίζονται απομακρυσμένα κατανεμημένα συστήματα καθόλη την έκταση του GENI

Το GENI υποστηρίζει ένα ευρύ φάσμα πειραματικών πρωτοκόλλων και τεχνικών διάδοσης δεδομένων, τα οποία λειτουργούν πάνω από υποδομές οπτικών ινών με οπτικούς μεταγωγείς, υψηλής ταχύτητας δρομολογητές, ασύρματα δίκτυα και clusters υψηλής υπολογισιμότητας.

4.3.1 Σχεδιαστικοί στόχοι

Οι στόχοι εξασφαλίζουν ότι η υποδομή στο σύνολό της θα είναι χρήσιμη στην ερευνητική κοινότητα ενθαρρύνοντας πειράματα μεγάλης κλίμακας με διαφορετικές και επεκτάσιμες τεχνολογίες. Αυτοί είναι οι εξής:

- Γενικότερα: το GENI θα πρέπει να δώσει σε κάθε ερευνητή την ευελιξία ώστε να εκτελέσει το επιθυμητό πείραμα.



- Διαφορετικότητα και επεκτασιμότητα: το GENI πρέπει να περιλαμβανει μια μεγάλη ποικιλία από δικτυακές τεχνολογίες, τόσο ενσύρματες όσο και ασύρματες. Θα πρέπει να είναι επεκτάσιμο, μέσα από αυστηρά καθορισμένες διαδικασίες, ώστε να ενσωματώσει νέες τεχνολογίες.
- Αξιοπιστία: το GENI θα πρέπει να επιτρέπει πειράματα τα οποία ανταποκρίνονται σε πραγματικά σενάρια και συνθήκες, καθώς και ισχυρή υποστήριξη για ποσοτικές μετρήσεις.
- ευκολία στη χρήση: το GENI θα πρέπει να αποτρέπει στο μεγαλύτερο βαθμό πρακτικά εμπόδια ώστε οι ερευνητές να κανουν πλήρη χρήση της υποδομής.
- Τμηματοποίηση: για να είναι οικονομικά αποδοτικό το GENI θα πρέπει να είναι μια διαμοιραζόμενη υποδομή, η οποία μπορεί να χρησιμοποιηθεί σε πολλαπλά πειράματα για λογαριασμό πολλών ανεξάρτητων ερευνητικών ομάδων.

- Ελεγχόμενη απομόνωση: το GENI πρέπει να υποστηρίζει απομόνωση μεταξύ των slices ώστε τα πειράματα να μην παρεμβάλλονται το ένα στο άλλο.
- Ασφάλεια: το GENI θα πρέπει να είναι ασφαλές ώστε οι πόροι του να μην μπορούν να χρησιμοποιηθούν εκούσια ή ακούσια για διαδικτυακές επιθέσεις.
- Υποστήριξη: το GENI πρέπει να σχεδιαστεί με χρονικό ορίζοντα λειτουργίας τα δεκαπέντε με είκοσι χρόνια.

4.3.2 Αρχιτεκτονική

Τα βασικά υποσυστήματα του GENI καθορίζονται με πολύ γενικό τρόπο ώστε να μπορούν να υποστηριχθούν όσο γίνεται πιο πολλές νέες τεχνολογίες και λειτουργικά μοντέλα. Αυτές οι οντότητες και τα χαρακτηριστικά τους περιγράφονται στη συνέχεια.

Σύνολα και Στοιχεία:
Το GENI περιλαμβάνει πολλά διαφορετικά σύνολα και στοιχεία, όλα ανεξάρτητα αλλά διαθέσιμα για πειράματα μέσω ενός πλαισίου που εκτελείται από το clearinghouse. Τα σύνολα δεν απαιτείται να είναι ομογενή, μπορεί να περιλαμβάνουν υπολογιστές, μεταγωγείς, δίκτυα αποθήκευσης δεδομένων κ.ά.. Τα στοιχεία συνήθως εικονικοποιούνται και είναι συχνά προγραμματίσιμα. Οι ερευνητές χρησιμοποιούν το GENI δεσμεύοντας πόρους, οι οποίοι μπορεί να είναι φυσικοί ή εικονικοί. Ο component manager παρέχει τη διεπαφή στο πλαίσιο ελέγχου, διαχειρίζεται την ανάθεση πόρων και παραμετροποιεί τα στοιχεία παρέχοντας slivers. Αντίστοιχα για τα σύνολα υπάρχει ένας aggregate manager.

Clearinghouse: Το clearinghouse είναι ένα λογισμικό το οποίο είναι κεντρικοποιημένο. Περιλαμβάνει principal, slice και component registries, με τις σχετικές υπηρεσίες. Επιπλέον το clearinghouse μπορεί να περιέχει τις παρακάτω προαιρετικές οντότητες.: ένα ticket log ή ένα software repository. Το principal registry κρατά αρχείο για κάθε ερευνητή που σχετίζεται με το GENI, για διαχειριστές συστημάτων, κλπ. Το slice registry κρατά αρχείο για κάθε slice, καταγράφοντας εξουσιοδοτημένες προσβάσεις και συναλλαγές. Το component registry κρατά αρχείο για κάθε στοιχείο ή σύνολο. Έτσι παρέχει πληροφορίες για όλα τα στοιχεία που είναι συμφωνημένα να συμμετέχουν σε πειράματα τα οποία χρησιμοποιούν το clearinghouse. Ένα ticket είναι μια εγγραφή sliver, η οποία καθορίζει τους πόρους που δεσμεύει ένα στοιχείο σε δεδομένο slice. Το clearinghouse έχει μια

υπηρεσία έκδοσης πιστοποιητικών σχετικά με το slice registry. Το ticket log κρατά αντίγραφα του αρχικού ticket (εγγραφή sliver) και τυχόν ενημερώσεις. Αυτό επιτρέπει στους διαχειριστές να βρίσκουν και να διαχειρίζονται όλα τα slivers που σχετίζονται με ένα συγκεκριμένο τμήμα του GENI, για παράδειγμα με ένα σύνολο από slices.

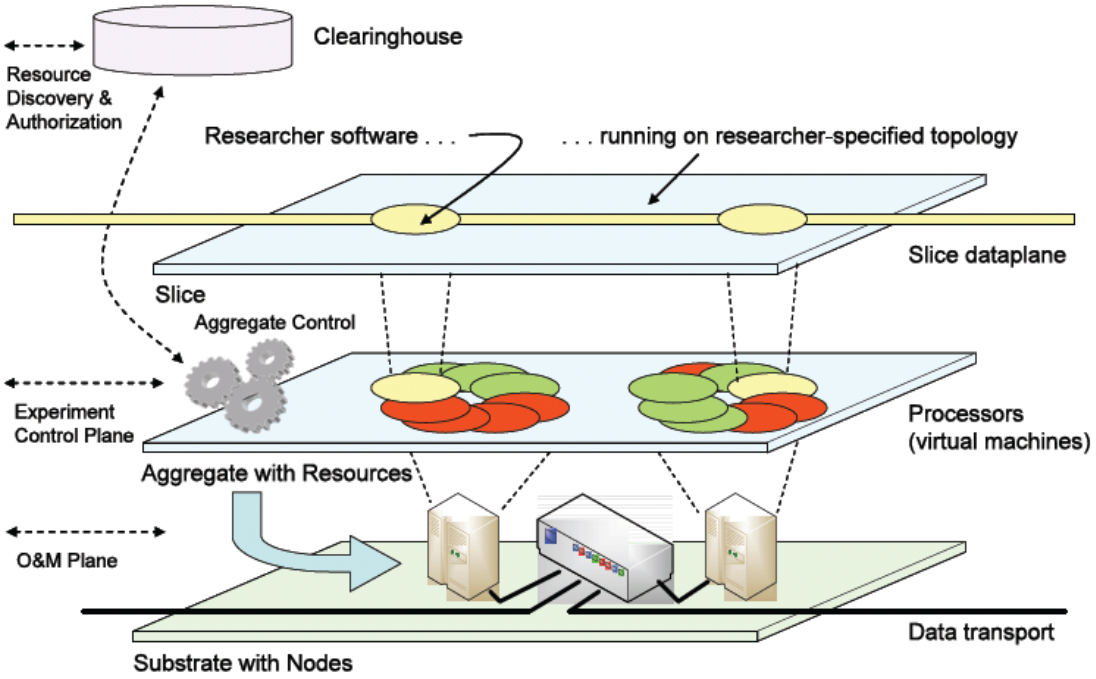
Οργανισμοί, ερευνητές και εργαλεία ελέγχου: Το GENI περιλαμβάνει ερευνητικούς οργανισμούς και ερευνητές που χρησιμοποιούν εργαλεία ελέγχου. Προαιρετικά local principal registries σε έναν ερευνητικό οργανισμό κρατούν στοιχεία για τους ερευνητές τους ρόλους τους και τις σχέσεις μεταξύ τους, με άλλους οργανισμούς και με ερευνητικά προγράμματα. Ένας σημαντικός ρόλος είναι οι slice administrators, που είναι υπεύθυνοι για τη δημιουργία των slice εγγραφών και εξουσιοδοτούν τους ερευνητές να εκτελέσουν πειράματα.

Υποστηρικτικές υπηρεσίες πειραμάτων: Αυτές οι υπηρεσίες περιλαμβάνουν υπηρεσίες αποθήκευσης λογισμικού, με τις οποίες οι ερευνητές αρχειοθετούν τις παραμετροποιήσεις και τα αποτελέσματα των πειραμάτων. Με τις Measurement Storage υπηρεσίες συλλέγουν τις πειραματικές μετρήσεις και υπηρεσίες εικονοποίησης όπου καταγράφεται η σχέση μεταξύ των ροών δεδομένων και των πειραμάτων.

Opt-in End Users: Οι Opt-in End Users είναι εκείνοι που επιλέγουν να συμμετέχουν σε ένα πείραμα. Οι χρήστες μπορεί να μην γνωρίζουν την ύπαρξη του GENI, αλλά να τρέχουν εφαρμογές που εκμεταλλεύονται τους πόρους του. Αυτό το γεγονός προσδίδει αξιοπιστία στα αποτελέσματα των πειραμάτων.

Λειτουργίες και διαχείριση: Παρέχονται από ανθρώπους που με κάποιο τρόπο εμπλέκονται σε τμήματα του GENI και διανέμονται ανάμεσα σε διάφορους οργανισμούς. Οι λειτουργίες και η διαχείριση παρέχουν επιπρόσθετα διεπαφές του GENI με τον έξω κόσμο όπως π.χ. το διαδίκτυο. Εδώ υλοποιείται το Help Desk.

Υποστρώματα, σύνολα και slices: Στο παρακάτω σχήμα φαίνονται οι συσχετίσεις μεταξύ των υποστρωμάτων, των συνόλων και των slices στο GENI.



- Ένα υπόστρωμα αποτελείται από το φυσικό εξοπλισμό και το λογισμικό διαχείρισης που το χειρίζεται. Κάποιος εξοπλισμός αυτού του επιπέδου δεν είναι ορατός στους ερευνητές.
- Ένα σύνολο αντιπροσωπεύει τα στοιχεία σε υψηλό επίπεδο, τα οποία βλέπει ο ερευνητής πιο γενικευμένα ως ένα ενιαίο σύνολο.
- Ένα slice είναι ένα κατανεμημένο δίκτυο από προγραμματίσιμα στοιχεία (εικονικοί επεξεργαστές, ζεύξεις κλπ.)

Στο υπόστρωμα βρίσκεται η ραχοκοκαλιά της υποδομής. Μπορεί να περιλαμβάνει τεχνολογίες όπως συνδεδεμένοι δρομολογητές μέσω MPLS συνδέσεων, ή μέσω IP tunnels ή Ethernet ή οπτικά δίκτυα. Μπορεί επίσης να αποτελείται από μη εμπορικό εξοπλισμό που βρίσκεται ακόμα σε στάδιο ανάπτυξης. Τα περισσότερα υποστρώματα περιλαμβάνουν εξοπλισμό που δεν φαίνεται στους ερευνητές ή αυτοι δεν μπορούν άμεσα να τον χειριστούν. Επιπλέον έχουν τα δικά τους εργαλεία διαχείρισης τα οποία συνήθως είναι αρκετά πολύπλοκα.

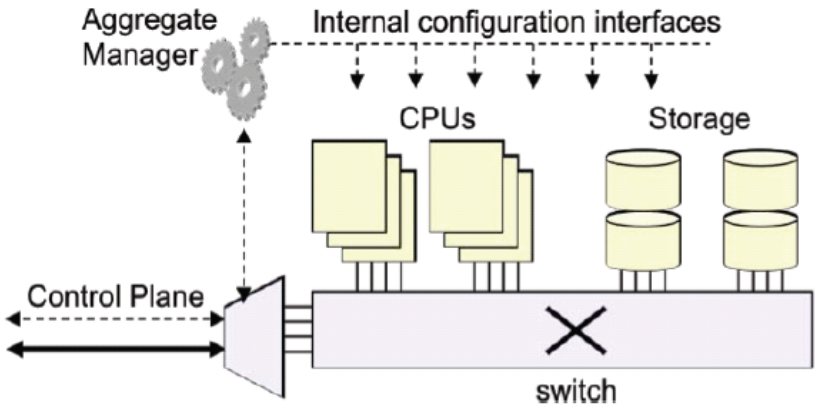
Στο επίπεδο του συνόλου κρύβονται τα πραγματικά χαρακτηριστικά του υποστρώματος και παρουσιάζεται ως ένα απλοποιημένο ενιαίο σύνολο στον ερευνητή. Εδώ βρίσκονται οι τυχόν εικονικές μηχανές.

Μόλις το slice εγκατασταθεί ο ερευνητής αποκτά τη δική του εικονική μηχανή στο δικό του εικονικό δίκτυο. Μέσα από το slice φαίνονται μόνο οι οντότητες που ανήκουν σε αυτό και δεν υπάρχει κάποια ένδειξη για τον έξω κόσμο.

4.3.3 Εφαρμογές του GENI

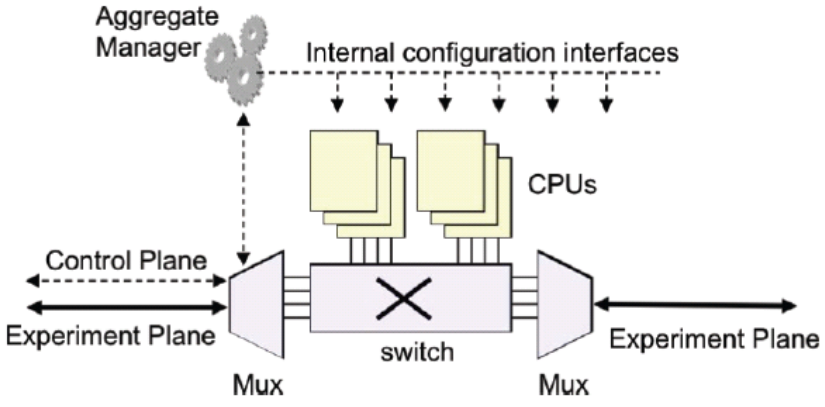
Πληθώρα από υποσυστήματα του GENI μπορούν να χαρτογραφηθούν και να απεικονισθούν σε ένα ενιαίο μοντέλο συνόλου.

CPU / storage clusters: Ένα CPU / Storage cluster φαίνεται στο παρακάτω σχήμα. Αποτελείται από επεξεργαστές και αποθηκευτικές συσκευές διασυνδεδεμένες μέσω ενός μεταγωγέα ή δικτύου μεταγωγής. Τα GENI slivers εκτελούν πειραματικό κώδικα στα CPUs μέσα σε ένα cluster και θα χρειαστούν πρόσβαση στο δικό τους τοπικό αποθηκευτικό χώρο.



Όταν ένα cluster αντιμετωπίζεται σαν ένα σύνολο ο ερευνητής μπορεί να απαιτήσει πόρους οι οποίοι περιλαμβάνουν CPUs, αποθηκευτικό χώρο και το διαχειριστής κάνει τις συνδέσεις ώστε να σχηματιστεί η ζητούμενη τοπολογία και απομόνωση για το slice. Ο μεταγωγέας του cluster είναι υπεύθυνος για την επικοινωνία αυτού με τον έξω κόσμο.

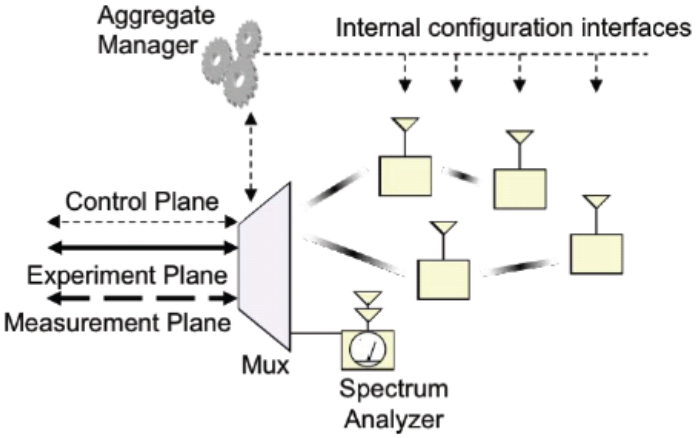
Προγραμματίσιμοι δρομολογητές: Στο παρακάτω σχήμα φαίνεται ένας δρομολογητής ως ένα σύνολο. Διαφέρει από το προηγούμενο σενάριο σε ΄δυο βασικά σημεία: α) ενσωματώνει πολλαπλές ζεύξεις μεταφοράς δεδομένων και β) ο κώδικας του ερευνητή μπορεί να εγκατασταθεί σε τμήματα του εξοπλισμού που κάνει πολυπλεξία για καλύτερη διαχείριση των ουρών.



Συγχρονισμένα σύνολα λογισμικού μπορούν να εγκατασταθούν στα CPUs και στους πολυπλέκτες έτσι ώστε ο κεντρικός μεταγωγέας δικτύου να υλοποιήσει ένα ξεχωριστό

VLAN για κάθε slice. Μερικοί ερευνητές μπορεί να επιθυμούν το εικονικό δικτύου να είναι τελείως απομονωμένο από κίνηση των άλλων slices ή κίνηση που δεν προέρχεται από το GENI ή σε ένα πιο ρεαλιστικό σενάριο τέτοια παρεμβολή να είναι επιθυμητή.

Ασύρματα Δίκτυα: Στο σχήμα φαίνεται ένα ασύρματο δίκτυο, το οποίο μπορεί να εκτείνεται σε ένα κτίριο ή σε μητροπολιτική έκταση. Εδώ οι κόμβοι συνδέονται με ραδιοζευξεις. Αν και μπορεί κάθε κόμβος να είναι τόσο μικρός ώστε να είναι δύσκολη η πολυπλεξία, ωστόσο τέτοια δίκτυα ταιριάζουν στο μοντέλο συνόλων του GENI.



Σε αυτούς τους τύπους συνόλων οι ερευνητές μπορεί να απαιτήσουν ή να ανακαλύψουν πόρους με βάση γεωγραφικά κριτήρια, μπορεί να εκτελέσουν πειράματα σε κινούμενους κόμβους ή μπορεί να ενδιαφέρονται για το φάσμα των ραδιοεκπομπών και των παρεμβολών. Κάθε τέτοιο δίκτυο ια χρειαστεί έναν GENI πολυπλέκτη, ο οποίος θα προσπαθήσει να μοιράσει δίκαια τη ζεύξη μεταξύ των απομονωμένων slices.

4.4 Federica

Το Federica [26] είναι μια δικτυακή υποδομή για πειράματα που δημιουργήθηκε έπειτα από συνεργασία ευρωπαϊκών πανεπιστημίων και ερευνητικών κέντρων με σκοπό την ανάπτυξη νέων δικτυακών τεχνολογιών και αρχιτεκτονικών για το μελλοντικό Internet αλλά και για virtual κατανεμημένα συστήματα. Ο δικτυακός εξοπλισμός και τα υπολογιστικά συστήματα από τα οποία αποτελείται το Federica υποστηρίζουν τεχνολογίες virtualization γεγονός που επιτρέπει τη δημιουργία virtual networks πάνω στη φυσική υποδομή. Ένας χρήστης μπορεί να αιτηθεί τη δημιουργία ενός slice, δεσμεύοντας μέρος των διαθέσιμων πόρων, που μπορεί να διαχειριστεί πλήρως εντός του οποίου θα εκτελέσει το πείραμα του. Κάθε slice είναι ένα virtual network, αποτελείται δηλαδή από virtual nodes και virtual links τα οποία μπορεί να υφίστανται παράλληλα και ανεξάρτητα με άλλα nodes και links που μπορούν να χρησιμοποιούνται σε άλλα πειράματα.

4.4.1 Στόχοι του Federica

Βασικός σκοπός του project είναι η κατανόηση και επινόηση καινοτόμων λύσεων για την παρακολούθηση, διαχείριση και τον έλεγχο παράλληλων virtual networks. Φιλοδοξία των δημιουργών του Federica είναι βοηθήσει στην έρευνα για τις τεχνολογίες και τις αρχιτεκτονικές που θα υιοθετηθούν στο μελλοντικό διαδίκτυο. Τα πειραματικά αποτελέσματα αναμένεται να διαμορφώσουν τις προδιαγραφές των μελλοντικών πρωτοκόλλων του διαδικτύου. Η χρήση των υποδομών του Federica αναμένεται ότι θα οδηγήσει στην βελτίωση της διαχείρισης των virtual υποδομών ως έναν συνδυασμό δικτύων και συστημάτων.

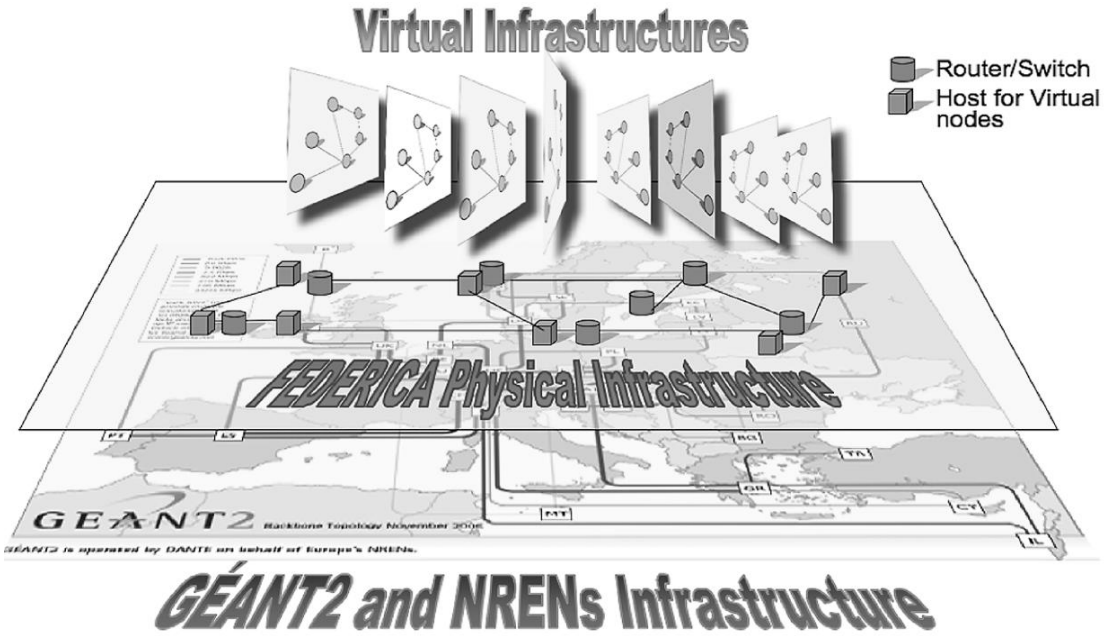
4.4.2 Προδιαγραφές

Κατά την σχεδίαση του Federica τέθηκαν κάποιες βασικές προδιαγραφές που παίζουν καθοριστικό ρόλο στην υλοποίηση του έργου και στην ανάδειξη ως καθοριστικού συντελεστή στην έρευνα για το μελλοντικό διαδίκτυο. Βασική αρχή κατά την σχεδίαση του Federica υπήρξε η υποστήριξη ταυτόχρονης εκτέλεσης πειραμάτων από πολλούς χρήστες χωρίς να επηρεάζεται το ένα από την εκτέλεση των υπολοίπων. Η απαίτηση αυτή συνεπάγεται την χρήση virtualization τεχνικών που εξασφαλίζουν απομόνωση (isolation) μεταξύ των διάφορων πειραμάτων. Επιπρόσθετα, πρέπει υπό τις ίδιες αρχικές συνθήκες που τίθεται σε ένα πείραμα να εξάγονται τα ίδια αποτελέσματα σε κάθε εκτέλεση ενός πειράματος αυτού. Επιπλέον πρέπει να υποστηρίζει οποιαδήποτε τεχνολογία να τρέξει σε ένα από τα virtual περιβάλλοντα που θα δημιουργηθούν πάνω σε αυτό. Σημαντικό είναι να παρέχει στον χρήστη πλήρη διαχειριστικό έλεγχο των πόρων που του έχουν ανατεθεί. Θα πρέπει να μπορεί να διασυνδεθεί (federate) με άλλες virtual υποδομές και το διαδίκτυο.

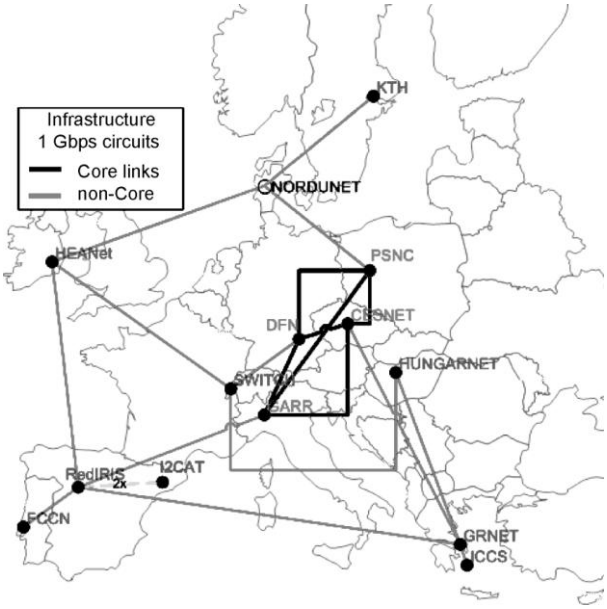
4.4.3 Αρχιτεκτονική του Federica

Το Federica αποτελείται από 2 διακριτά επίπεδα. Τη φυσική υποδομή που περιλαμβάνει το hardware και το software που υποστηρίζουν τη χρήση virtualization και το επίπεδο που περιλαμβάνει τις εικονικές δικτυακές υποδομές (slices). Το συγκεκριμένο επίπεδο έχει δική του ξεχωριστή διαχείριση. Το πλήθος των εικονικών δικτυακών υποδομών (slices) είναι θεωρητικά απεριόριστο, στην πραγματικότητα όμως επηρεάζεται από τους διαθέσιμους φυσικούς πόρους και τις απαιτήσεις που υπάρχουν για κάθε slice. Όταν η εικονικοποίηση δεν είναι διαθέσιμη στο υλικό όπως σε περιπτώσεις virtualization των δικτυακών διεπαφών

ενός υπολογιστή χρειάζεται η εγκατάσταση επιπλέον υλικού. Παρακάτω παρουσιάζονται τα δύο επίπεδα του Federica.



Το Federica υποστηρίζει διασύνδεση με άλλες υπάρχουσες δικτυακές υποδομές καθώς και με το διαδίκτυο. Αποτελείται από slices ώστε οι υποδομές του να μπορούν να χρησιμοποιηθούν ταυτόχρονα από πολλές ερευνητικές ομάδες για την ανάπτυξη διαφορετικών υποδομών πλήρως απομονωμένων χάρις στην χρήση virtualization τεχνικών . Κάθε slice μπορεί να παραμετροποιηθεί ανάλογα με τις ερευνητικές ανάγκες που προορίζεται να καλύψει. Το Federica έχει υλοποιηθεί πάνω σε υβριδικά δίκτυα έπειτα από συνεργασία ευρωπαϊκών ερευνητικών και εκπαιδευτικών οργανισμών (National Research and Education Networks – NREN) με το GEANT2 το DANTE το TERENA και άλλα ακαδημαϊκά και βιομηχανικά group. Τα σημεία παρουσίας (Points of Presence – PoPs) βασίζονται στη χρήση εξοπλισμού μεταγωγής και δρομολόγησης που υποστηρίζει virtualization και διασυνδέονται μεταξύ τους με ζεύξεις Ethernet και MPLS tunnels πάνω από το GEANT2 ή άλλα ερευνητικά δίκτυα. Τα κεντρικά PoPs βρίσκονται στην Τσεχία, στην Γερμανία την Ιταλία και την Πολωνία. Άλλα εννέα σημεία παρουσίας θα διασυνδεθούν σταδιακά στις υπόλοιπες συμμετέχουσες στο έργο χώρες.



Οι τέσσερις κεντρικοί κόμβοι (core nodes) του Federica αποτελούνται από MX routers/switches ενώ οι υπόλοιποι εννέα από EX switches. Στους τέσσερις κεντρικούς κόμβους τρέχουν 2 V-Nodes και από ένας σε κάθε non-core κόμβο.

Το δίκτυο αποτελεί ένα αυτόνομο σύστημα με δημόσιες και ιδιωτικές διευθύνσεις IPv4 και IPv6. Η διασύνδεση του δικτύου του Federica με το διαδίκτυο γίνεται με χρήση του πρωτοκόλλου BGP μέσω των τεσσάρων κεντρικών PoPs στα routers των οποίων εγγράφονται οι αντίστοιχες καταχωρήσεις του BGP. Η διαχείριση του γίνεται κεντρικά από ένα κέντρο διαχείρισης λειτουργιών (Network Operation Center – NOC) που είναι αρμόδιο για την δημιουργία των εικονικών δικτυακών υποδομών (slices). Η φυσική υποδομή του Federica αποτελείται από τη διασύνδεση διάφορων ευρωπαϊκών ερευνητικών και εκπαιδευτικών δικτύων (NRENs) με το δίκτυο κορμού του GEANT2 Για το virtualization των κόμβων χρησιμοποιήθηκε το περιβάλλον VMware και η έκδοση ESX. Στην παρούσα φάση ανάπτυξης του Federica το data link layer υλοποιείται μέσω Ethernet framing. Η μελλοντική ανάπτυξη της υποδομής ίσως επιτρέψει πρόσβαση στον οπτικό εξοπλισμό προς υπέρβαση αυτού του περιορισμού.

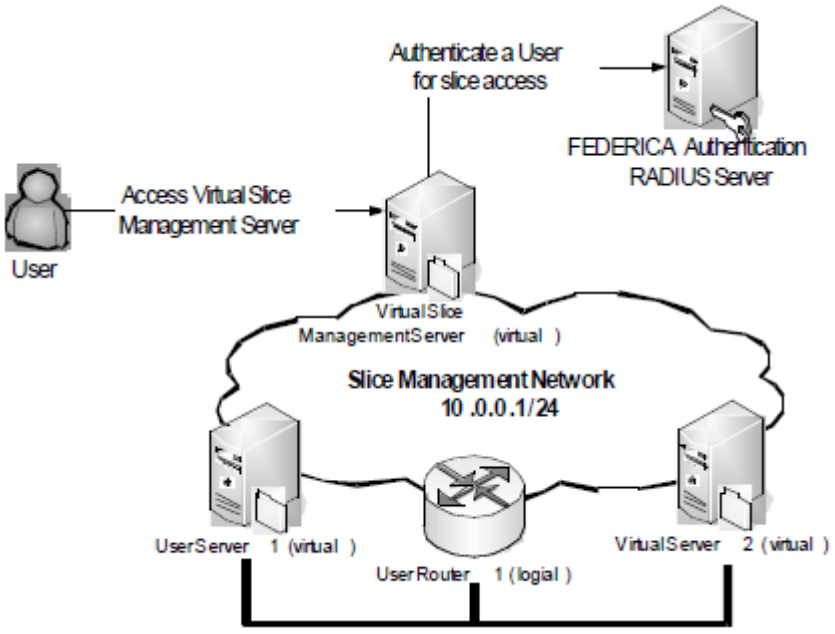
4.4.4 Διαδικασία δημιουργίας slices – Δέσμευση πόρων

Πρώτο στάδιο ανάπτυξης

Η δημιουργία ενός virtualized δικτυακού περιβάλλοντος στην υποδομή του Federica συνίσταται στους V-nodes και στα virtual links μεταξύ των κόμβων του δικτύου. Όταν έναν

ερευνητής επιθυμεί να εκτελέσει ένα πείραμα στέλνει αίτημα προς το NOC με την επιθυμητή τοπολογία και τους πόρους (μνήμη RAM, ισχύς CPU, χώρο αποθήκευσης, εύρος ζώνης για τις ζεύξεις μεταξύ των V-nodes) που επιθυμεί να δεσμεύσει με σκοπό τη δημιουργία ενός slice στο οποίο θα μπορεί να εκτελεστεί το πείραμα πλήρως απομονωμένα σε σχέση με άλλα που μπορούν να εκτελούνται παράλληλα σε άλλα slices. Όταν το NOC λάβει το αίτημα για το slice γίνεται επιλογή από ποια σημεία παρουσίας θα δεσμευτούν οι αιτούμενοι πόροι αντιστοιχίζοντας έτσι την αιτούμενη εικονική τοπολογία με φυσική τοπολογία. Η επικοινωνία του slice με το διαδίκτυο γίνεται μέσω μιας virtual machine που δημιουργείται από το NOC και λειτουργεί ως gateway μεταξύ του internet και του slice και ορίζεται ως Slice Management Server. Ο έλεγχος πρόσβασης στον Slice Management Server γίνεται μέσω identity credentials μέσω ενός RADIUS server. Η δημιουργία της αιτούμενης τοπολογίας υλοποιείται μέσω Ethernet VLANs. Η διαχείριση του slice γίνεται μέσω ενός Slice Management Network. Κάθε πόρος του slice (V-nodes, logical routers, software routers) διασυνδέεται μέσω εικονικών διεπαφών με την εικονική διεπαφή του Management Server. Οι πόροι και ο Management Server ανήκουν στο ίδιο υποδίκτυο αποτελούμενο από ιδιωτικές IP και στο ίδιο VLAN. Το υποδίκτυο αυτό είναι ο τρόπος πρόσβασης τους πόρους του slice όταν κάποιος συνδέεται με τον Management Server μέσω του διαδικτύου.

Όταν δημιουργηθεί το slice το NOC στέλνει στον ερευνητή την δημόσια IP διεύθυνση του Virtual Slice Management Server και τους κωδικούς πρόσβασης σε αυτόν (credentials), και το IP addressing scheme του Virtual Slice Management Network.



Στο παραπάνω παράδειγμα διαθέτουμε ένα slice που αποτελείται από 2 virtual servers και ένα Juniper στον ίδιο VLAN. Ο ερευνητής μπορεί να συνδεθεί με τον Virtual Slice Management Server και μέσω αυτού να διαχειριστεί το slice.

Δεύτερο στάδιο ανάπτυξης

Ένα μεγάλο ζήτημα που προκύπτει από τα παραπάνω όσο αφορά στην λειτουργία του Federica ως μιας εικονικής υποδομής για πειράματα σχετίζεται με την διαδικασία δημιουργία ενός slice και την δέσμευση των πόρων. Η διαδικασία, όπως περιγράφηκε και προηγουμένως γίνεται έπειτα από αίτημα στο NOC με μη αυτοματοποιημένο τρόπο. Έπειτα από ερευνητικές προσπάθειες, επιτεύχθηκε η αντιστοίχιση μεταξύ φυσικής και virtual τοπολογίας, που συνίσταται στην δέσμευση των πόρων και τον ορισμό των virtual links, να γίνεται με αυτοματοποιημένο τρόπο τέτοιο ώστε να είναι σύμφωνος με το Service Level Agreement. Πιο συγκεκριμένα, με την ανάπτυξη του Federica Slice tool [27] και του User Portal επιτεύχθηκε αυτοματοποίηση στη δημιουργία και κατανομή των slices καθώς και διαδικασία εγγραφής των χρηστών σε κάθε slice. Το portal που αναπτύχθηκε για τους χρήστες του Federica τους δίνει την δυνατότητα να αιτούνται τη δημιουργία νέων slices ενώ το Federica Slice Tool λειτουργεί ως επίπεδο διαχείρισης (management plane).

4.5 PlanetLAb

Το PlanetLab [9] είναι ένα κατανεμημένο δίκτυο επικάλυψης με σκοπό την υποστήριξη και αποτίμηση μεγάλης κλίμακας δικτυακών υπηρεσιών. Αποτελείται από 350 machines σε 150 τοποθεσίες και 20 χώρες και υποστηρίζει περισσότερα από 450 ερευνητικά έργα. Το PlanetLab παρέχει distributed virtualization ώστε να μπορεί να διαθέτει την υποδομή του προς ταυτόχρονη χρήση σε πολλές ερευνητικές ομάδες, επιτρέποντας έτσι την παροχή κάθε υπηρεσίας σε ξεχωριστό απομονωμένο slice που προκύπτει από δέσμευση πόρων κατά μήκος του δικτύου. Επιπλέον, με σκοπό την ενίσχυση του ανταγωνισμού μεταξύ των προσφερόμενων υπηρεσιών μέσω του δικτύου, παρέχεται η δυνατότητα αποδέσμευσης της παρεχόμενης υπηρεσίας από το λειτουργικό σύστημα το οποίο τρέχει σε έναν κόμβο (unbundled management).

4.5.1 Προδιαγραφές

Το Planetlab είναι κατανεμημένο σύστημα γεγονός που σημαίνει ότι διέπεται από συγκεκριμένες σχέσεις μεταξύ των διαχειριστών των κόμβων, των χρηστών, των ερευνητών και το παροχέων υπηρεσιών. Οι σχέσεις που πρέπει να καθοριστούν ώστε να ληφθούν υπόψη κατά την σχεδίαση του PlanetLab είναι οι ακόλουθες:

- Μεταξύ του Planetlab ως οργανισμού και των ερευνητικών κέντρων και πανεπιστημίων που κατέχουν τους κόμβους που χρησιμοποιούνται για την ανάπτυξη του δικτύου του. Ο οργανισμός που διαχειρίζεται το PlanetLab έχει διαχειριστικό έλεγχο στους κόμβους αυτούς, όμως και τα πανεπιστήμια μπορούν να καθορίζουν πολιτικές που αφορούν τον τρόπο χρήσης των κόμβων, τα είδη και την ποσότητα της δικτυακής κίνησης που μπορούν να παράγουν. Κατά συνέπεια χρειάζεται να διαμοιραστεί ο έλεγχος στους κόμβους του PlanetLab.
- Μεταξύ του PlanetLab και των χρηστών που το χρησιμοποιούν. Πρόκειται για ερευνητές που επιθυμούν να αποτιμήσουν μεγάλης κλίμακας υπό σχεδίαση υπηρεσίες. Οι χρήστες πρέπει να έχουν πρόσβαση την πλατφόρμα του PlanetLab από κάθε κόμβο γεγονός που συνεπάγεται την δημιουργία ενός slice μέσω χρήσης distributed virtualization
- Μεταξύ του PlanetLab και των ερευνητών που σχεδιάζουν υπηρεσίες υποδομής όπως το VINI. Οι υπηρεσίες αυτές που παρέχονται από πολλές διαφορετικές ερευνητικές ομάδες θα πρέπει να υφίστανται σε ξεχωριστά slices που να υπάρχουν παράλληλα και να είναι πλήρως απομονωμένα μεταξύ τους.
- Μεταξύ του Planetlab και του υπόλοιπου internet καθώς πρέπει να ληφθεί υπόψη ότι τα πειράματα που θα λαμβάνουν χώρα στο PlanetLab ενδέχεται να επιδρούν αρνητικά στο υπόλοιπο διαδίκτυο. Πρέπει να ληφθεί μέριμνα σχετικά με το είδος της τηλεπικοινωνιακής κίνησης που οι χρήστες επιτρέπεται να διακινούν στο υπόλοιπο διαδίκτυο.

Το PlanetLab αναπτύσσεται στην λογική των slices. Σε κάθε slice μπορεί να παρέχεται μια συγκεκριμένη υπηρεσία. Σε κάθε slice διατίθεται μέρος των πόρων κάθε κόμβου υπό την μορφή virtual machines. Σε αντίθεση με την απλή virtualization που υφίσταται σε άλλα ερευνητικά projects, στο PlanetLab έχουμε distributed virtualization που σημαίνει ότι πλήθος virtual machines που υφίστανται σε διαφορετικούς φυσικούς κόμβους νοούνται ως

μια ενιαία οντότητα από το σύστημα. Για την υλοποίηση της distributed virtualization θα πρέπει να διασφαλίζονται δύο βασικά χαρακτηριστικά σχετικά με τα slices:

4.5.2 Απομόνωση των slices

Θα πρέπει να διασφαλίζεται απομόνωση των slices λαμβάνοντας υπόψη ότι κάθε ένα από αυτά συνδέει distributed ομάδες που η κάθε μια αποτελείται από ένα πλήθος virtual machines. Επιπλέον, η δέσμευση των πόρων σε κάθε κόμβο θα πρέπει να γίνεται με τέτοιο τρόπο ώστε να εξασφαλίζεται η ελάχιστη ποιότητα υπηρεσίας που απαιτείται για την εφαρμογή που θα εκτελεστεί στο συγκεκριμένο slice. Θα πρέπει να ληφθούν υπόψη τρεις παράγοντες για την σχεδίαση του λειτουργικού συστήματος του PlanetLab:

- Πρέπει να κατανέμει και να προγραμματίζει τους πόρους των κόμβων με τέτοιο τρόπο ώστε να μην επηρεάζεται η επίδοση των υπηρεσιών που υφίστανται σε άλλα slices στον ίδιο κόμβο. Ορισμένα slices θα πρέπει να μπορούν να ζητάνε ένα ελάχιστο επίπεδο πόρων ώστε να εξασφαλίζουν έτσι μια συγκεκριμένη επίδοση οι υπηρεσίες που υφίστανται σε αυτά.
- Πρέπει να γίνεται κατάτμηση των διαθέσιμων namespaces όπως είναι οι δικτυακές διευθύνσεις και τα ονόματα των αρχείων ώστε να εξασφαλίζεται ότι κάθε slice δεν έχει πρόσβαση σε πληροφορίες των υπολοίπων. Η απαίτηση αυτή ουσιαστικά συνίστανται στην απομόνωση που πρέπει να υπάρχει μεταξύ των slices.
- Πρέπει η δομή του λειτουργικού συστήματος να είναι τέτοια ώστε ο κώδικας που αναφέρεται στην υπηρεσία ενός slice να μην έχει επίδραση στις ομαλή εκτέλεση των υπηρεσιών στα υπόλοιπα slices με την έννοια ότι δεν θα υπάρχει κάποια ευνοϊκότερη μεταχείριση στις εντολές που απαιτούνται να εκτελεστούν από ένα slice σε σχέση με ένα άλλο.

4.5.3 Απομόνωση του PlanetLab

Η απομόνωση του PlanetLab συνίσταται στην απαίτηση τα slices του PlanetLab να μην επιδρούν αρνητικά στο υπόλοιπο διαδίκτυο. Εξάλλου οι κόμβοι του PlanetLab διασυνδέονται με πολλαπλές λογικές συνδέσεις μέσω του διαδικτύου. Ελλοχεύει ο κίνδυνος κάποια υπηρεσία με σφάλματα (bugs) να έχει επιπτώσεις στην εύρυθμη λειτουργία του διαδικτύου λαμβάνοντας υπόψη και την μεγάλης κλίμακας υποδομή του PlanetLab. Για την

απομόνωση των slices του PlanetLab από το υπόλοιπο διαδίκτυο τίθενται οι εξής δύο προδιαγραφές για το λειτουργικό του σύστημα:

- Πρέπει να θέτει όρια στην κατανάλωση των πόρων από συγκεκριμένες υπηρεσίες ώστε να μην έχουμε επιπτώσεις στον ίδιο τον κόμβο που φιλοξενεί την υπηρεσία, όπως για παράδειγμα στην περίπτωσης της δέσμευσης αρκετού bandwidth που έχει επίπτωση σε άλλες υπηρεσίες εκτός PlanetLab που παρέχονται από τον κόμβο αυτό, αλλά και σε απομακρυσμένους κόμβους που δέχονται κίνηση από το κόμβο. Οι παραπάνω περιορισμοί πρέπει να τίθενται σε συνεργασία με τον διαχειριστή κάθε κόμβου και από το PlanetLab ως οργανισμό.
- Πρέπει να είναι εύκολη η παρακολούθηση χρήσης των πόρων. Η γνώση του πως οι χρήστες ή οι υπηρεσίες τους επηρεάζουν τον εξωτερικό κόσμο είναι κάτι καινούριο για τα συστήματα διαμοιραζόμενου χρόνου.

Η απομόνωση του PlanetLab προσφέρει ασφάλεια που ήταν από την αρχή ένα σημαντικό στοιχείο για το σχεδιασμό του. Παρόλα αυτά, ο αποτελεσματικός περιορισμός των χρηστών και η παρακολούθησή τους είναι το αρκετά σοβαρές παράμετροι. Για παράδειγμα, ένας χρήστης του PlanetLab που εκτελεί πειράματα διαπερατότητας TCP σε κόμβους του U.C. Berkeley κατάφερε να δεσμεύσει πάνω από το μισό διαθέσιμο εύρος ζώνης μέσα σε λίγες μέρες. Επίσης, πολλά πειράματα, όπως η χαρτογράφηση του διαδικτύου ενεργοποίησε μηχανισμούς IDS με αποτέλεσμα τοπικοί διαχειριστές να κλείσουν κόμβους.

4.5.4 Unbundled management

Ένας βασικός στόχος του PlanetLab είναι η παροχή πληθώρας υπηρεσιών μέσω των slices που αναπτύσσονται υπό την μορφή λογικών συνδέσεων στην υποδομή του που περιλαμβάνει ένα μεγάλο τμήμα του διαδικτύου. Ο τελικός χρήστης μπορεί να επιλέξει όποια από τις παρεχόμενες υπηρεσίες επιθυμεί. Η φιλοσοφία του unbundled management επιτάσσει οι παρεχόμενες από το PlanetLab υπηρεσίες υποδομών να μπορούν να παρέχονται και να αναπτύσσονται παράλληλα. Ένα ζήτημα που ανακύπτει όσο αφορά την εξέλιξη των υποδομών σχετίζεται με τη δυνατότητα παράλληλης χρήσης δύο εκδόσεων ενός συστήματος. Για την διασφάλιση του unbundled management στη πλατφόρμα του PlanetLab υπάρχουν δύο απαιτήσεις:

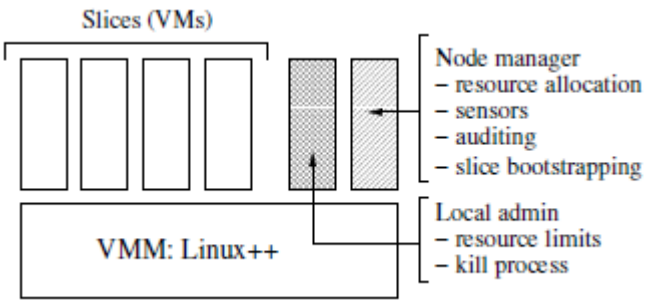
- Πρέπει να μειωθούν οι λειτουργίες του λειτουργικού σύστημα του PlanetLab και παράλληλα να αυξηθούν οι παρεχόμενες από τις υπηρεσίες λειτουργίες. Με αυτόν τον

τρόπο μόνο τοπικές λειτουργίες, λειτουργίες δηλαδή που αφορούν τον κόμβο, θα εκτελούνται από το λειτουργικό σύστημα ενώ οι υπόλοιπες από τις παρεχόμενες υπηρεσίες που εκτελούνται στα slices.

- Για την διασφάλιση υγιούς ανταγωνισμού μεταξύ των παρεχόμενων υπηρεσιών θα πρέπει η διεπαφή μεταξύ των υπηρεσιών υποδομής και του λειτουργικού συστήματος είναι διαμοιραζόμενη χωρίς ειδικά προνόμια.

4.5.5 Αρχιτεκτονική του PlanetLab

Η αρχιτεκτονική του PlanetLab αναπτύχθηκε με τέτοιο τρόπο ώστε να παρέχεται η δυνατότητα του unbundled management. Στο χαμηλότερο επίπεδο σε κάθε κόμβο του PlanetLab εκτελείται ένα virtual machine monitor (VMM) που υλοποιεί και απομονώνει τις virtual machines. Η υλοποίηση του VMM βασίζεται σε συνδιασμό του πυρήνα του Linux και κατάλληλων επεκτάσεων αυτού. Πάνω από το VMM τρέχει μία privileged virtual machine που ονομάζεται node manager και είναι υπεύθυνη για την παρακολούθηση και την διαχείριση των virtual machines που τρέχουν στον συγκεκριμένο κόμβο. Ο node manager είναι υπεύθυνος για την εφαρμογή συγκεκριμένος πολιτικών στις virtual machines καθώς και για την κατανομή πόρων σε αυτές. Ο node manager δέχεται επίσης αιτήματα από τις υπηρεσίες για δημιουργία νέων virtual machines κατά προτεραιότητα σε σχέση με το VMM. Τα αιτήματα αυτά έχουν τοπικό χαρακτήρα με την έννοια ότι μόνο υπηρεσίες που παρέχονται από virtual machines ενός κόμβου μπορούν να αιτηθούν και κατά συνέπεια απομακρυσμένη πρόσβαση στον node manager από άλλον κόμβο μπορεί να γίνει μόνο έμμεσα μέσω υπηρεσιών που τρέχουν στον κόμβο. Μελλοντικά ο διαχειριστής του κόμβου θα έχει περισσότερες δυνατότητες παραμετροποίησης όσο αφορά την επιβολή κάποιων κανόνων στον κόμβο διαχείρισης.



Ένα υποσύνολο των παρεχόμενων υπηρεσιών μέσω του PlanetLab, δηλαδή μερικά από τα slices που τρέχουν από τον VMM μπορούν να θεωρηθούν privileged όταν τους επιτρέπεται να έχουν privileged κλήσεις προς τον node manager. Η πλειοψηφία των slices είναι unprivileged όμως μερικές υπηρεσίες υποδομής (infrastructure services) χρειάζεται να τρέχουν σε privileged slice. Οι υπηρεσίες υποδομής που πρέπει να έχουν privileged δικαιώματα όσο αφορά την πρόσβαση τους στον node manager είναι οι ακόλουθες:

- Brokerage services που χρησιμοποιούνται για την δέσμευση πόρων και την δημιουργία slices
- Environment services που χρησιμοποιούνται για να εκκινούν και να συντηρούν τον κώδικα του slice
- Monitoring services που χρησιμοποιούνται με σκοπό την ανεύρεση των απαραίτητων πόρων καθώς και για την παρακολούθηση των υπηρεσιών και κατεπέκταση τη διασφάλιση της εύρυθμης λειτουργίας των slices.

Λόγω ενσωμάτωσης μελλοντικά νέων λειτουργιών στην αρχιτεκτονική του PlanetLab εγείρεται ζήτημα σχετικά με το που θα πρέπει να υλοποιηθούν σε privileged, unprivileged slices, στον node manager ή στον VMM. Η απόφαση θα πρέπει να ληφθεί αφού εκτιμηθούν οι ακόλουθοι δύο παράγοντες:

- Κάθε νέα λειτουργία θα πρέπει να υλοποιηθεί στο υψηλότερο δυνατό επίπεδο. Για παράδειγμα το να τρέχει μια υπηρεσία σε ένα slice με περιορισμένα privileged δικαιώματα είναι προτιμότερο από ένα slice με διευρυμένα δικαιώματα.
- Σε privileged slice θα πρέπει να εκχωρηθούν μόνο τα ελάχιστα απαιτούμενα δικαιώματα για την εύρυθμη λειτουργία της υποστηριζόμενης υπηρεσίας.

4.5.6 Federation στο PlanetLab

Βασικό στοιχείο για την διασφάλιση του federation στο PlanetLab αποτελεί η δυνατότητα των χρηστών να μπορούν να προσθέτουν κόμβους από πολλαπλές διαχειριστικές αρχές – MA στα slices τους. Αυτό διασφαλίζεται διότι ένα SA μπορεί να επικοινωνεί άμεσα με κόμβους που δεν υπάγονται υπό τον έλεγχο της ίδιας MA. Παρόλο που οι χρήστες έχουν τη δυνατότητα να χρησιμοποιούν πόρους από πολλαπλά PlanetLab συστήματα, ένα σημαντικό θέμα που τίθεται αφορά την ετερογένεια στη τεχνολογία που χρησιμοποιούν τα διάφορα

Planetlab συστήματα. Το πρόβλημα αυτό της ετερογένειας επιλύεται μέσω της κατανεμημένης εικονικοποίησης και του unbundled management.

4.6 ORBIT

Η ανάπτυξη μιας ασύρματης πειραματικής πλατφόρμας πολλαπλών χρηστών θέτει ζητήματα και προκλήσεις που δεν εμφανίζονται σε testbeds ενσύρματων δικτυων, όπως για παράδειγμα το Emulab και το ABone. Πιο συγκεκριμένα είναι πολύ πιο δύσκολο να εγκατασταθεί ένα πείραμα ασύρματου δικτύου λόγω των τυχαίων χρονικών αποκλίσεων στις τοποθεσίες των κινούμενων χρηστών και των συσχετιζόμενων μοντέλων ασύρματων καναλιών. Επιπρόσθετα, τα ασύρματα συστήματα τείνουν να παρουσιάζουν πολύπλοκες διαδράσεις μεταξύ του φυσικού ελέγχου πρόσβασης του μέσου και των επιπέδων δικτύου, με τέτοιο τρόπο ώστε προσεγγίσεις που χρησιμοποιούνται για να απλουστεύσουν τα πρότυπα ενσύρματων δικτύων να μην μπορούν να εφαρμοστούν στη δεδομένη περίπτωση. Κάποια χαρακτηριστικά των ραδιοκαναλιών που θα πρέπει να ενσωματωθούν σε ένα testbed ασύρματου δικτύου είναι τα ακόλουθα:

- Οι ιδιότητες των ραδιοκαναλιών εξαρτώνται από τις τοποθεσίες των ασύρματων κόμβων
- Οι ρυθμοί λαθών bit εξαρτώνται από το χρόνο
- Τα layer-2 πρωτόκολλα διαμοιραζόμενου μέσου στη ραδιοζεύξη έχουν ισχυρή επίδραση στην απόδοση του δικτύου
- Υπάρχουν πολύπλοκες αλληλεπιδράσεις των στρωμάτων της στοίβας στο ασύρματο πρωτόκολλο, οι οποίες δεν μπορούν να μελετηθούν εύκολα
- Η τυχαιότητα στην κίνηση και θέση του χρήστη δε μπορεί να θεωρηθεί αμελητέα

Ένα ευέλικτο testbed για ασύρματα δίκτυα θα πρέπει να μπορεί να υποστηρίξει πειραματική έρευνα σε διάφορες τοπολογίες δικτύου και δικτυακά πρωτόκολλα. Για να είναι αυτό χρήσιμο θα πρέπει να είναι επεκτάσιμο και να αντιμετωπίζει ένα μεγάλο φάσμα ερευνητικών θεμάτων που αφορούν τα ασύρματα δίκτυα.
Μερικά παραδείγματα συστημάτων και πρωτοκόλλων που θα μπορούσαν να ληφθούν υπόψη είναι:

- Μεγάλης κλίμακας ασύρματα δίκτυα βασισμένα στα 802.11a/b/g πρότυπα μαζί με νέα πρωτόκολλα εύρεσης, δρομολόγησης, ασφάλειας κ.α.

- MANETs (Mobile ad hoc networks) που τυπικά βασίζονται στα 802.11x πρότυπα, και επεκτείνονται για να υποστηρίξουν πρωτόκολλα δρομολόγησης όπως το AODVR (Ad-hoc On-Demand Distance Vector Routing) και DSR (Dynamic Source Routing)
- Κινητές εφαρμογές όπως υπηρεσίες που βασίζονται στην τοποθεσία, VoIP over MANET κ.α.

Η αρχιτεκτονική του συστήματος του ORBIT [15] testbed βασίζεται στις γενικές απαιτήσεις που περιγράφθηκαν στις προηγούμενες γραμμές. Οι βασικοί σχεδιαστικοί στόχοι είναι:

- επεκτασιμότητα, αναφορικά με το συνολικό αριθμό ασύρματων κόμβων
- αναπαραγωγή των πειραμάτων σε παρόμοια περιβάλλοντα και έκδοση παρόμοιων αποτελεσμάτων
- ευελιξία ανοιχτής πρόσβασης που δίνει μεγάλο έλεγχο πρωτοκόλλων και λογισμικού που χρησιμοποιείται στους κόμβους.
- testbed απομακρυσμένης πρόσβασης ικανό να χειρίζεται κατάλληλα τυχόν αποτυχίες υλικού και λογισμικού

Το testbed ORBIT χρησιμοποιεί αρχιτεκτονική two-tier με ένα εργαστηριακό προσομοιωτή ώστε να είναι δυνατή η αναπαραγωγή του πειράματος και ταυτόχρονα μπορεί να αξιολογεί την απόδοση του πρωτοκόλλου και της εφαρμογής σε πραγματικές συνθήκες. Χρησιμοποιεί μια δισδιάστατη διάταξη από στατικούς κόμβους 802.11x , οι οποίοι μπορούν δυναμικά να διασυνδεθούν σε συγκεκριμένες τοπολογίες για πειράματα. Μόλις το πρωτόκολλο ή η εφαρμογή αξιολογηθούν και στον εργαστηριακό προσομοιωτή, οι χρήστες μπορούν να μεταφέρουν το λογισμικό σε ένα δοκιμαστικό δίκτυο που παρέχει παραμετροποιήσιμη πρόσβαση μέσω κυψελωτού δικτύου $3^{ης}$ γενιάς και 802.11x σε πραγματικές συνθήκες . Εργαλεία συλλογής πειραματικών δεδομένων παρέχονται για να αξιολογήσουν την δικτυακή κίνηση, την απόδοση, την ποιότητα της ραδιοζεύξης και τη χρήση του φάσματος. Κάθε ραδιοκόμβος του ORBIT αποτελείται από:

- έναν επεξεργαστή 1GHz με 512MB μνήμη και 20GB σκληρό δίσκο.
- από δύο ασύρματες διεπαφές 802,11a/b/g
- δύο θύρες Ethernet 100Mbps
- έναν ολοκληρωμένο chassis manager που χρησιμεύει για να παρακολουθείται απομακρυσμένα η κατάσταση του κάθε κόμβου. Οι κόμβοι μπορούν να απενεργοποιηθούν ή να επανεκκινηθούν μέσω απομακρυσμένης διαχείρισης μέσω μιας τρίτης διεπαφής Ethernet

Ένα υποσύστημα παρέχει μετρήσεις για τα επίπεδα των ραδιοσημάτων και δημιουργεί διάφορων τύπων παρεμβολές RF (π.χ. λευκός θόρυβος) μέσα στο πλέγμα των κόμβων. Υπάρχει επίσης ένα ανεξάρτητο σύστημα παρακολούθησης WLAN το οποίο παρέχει μια οπτική σε επίπεδο δικτύου/MAC χρησιμοποιώντας παρατηρητές WLAN που βρίσκονται διασκορπισμένοι στο σύστημα. Τέλος, υπάρχουν κάποιοι εξυπηρετητές υποστήριξης, οι μεν front-end παρέχουν web services, οι δε back-end χρησιμοποιούνται για πειράματα και αποθήκευση δεδομένων.

Από πλευράς λογισμικού για τη διαχείριση και τον έλεγχο αναπτύχθηκαν τα εξής στοιχεία:

1. Node Handler. Σκοπός του είναι η διάδοση των πειραματικών σεναρίων μέσω multicast στους Node Agents ώστε να διεξαχθούν τα πειράματα και οι μετρήσεις. Οι Node Agents την κατάσταση της εκτέλεσης της εντολής στον Node Handler.
2. Collection Server. Σκοπός του είναι να συλλέγει τις μετρήσεις και τα στατιστικά των πειραμάτων από τους κόμβους.
3. Disk-Loading Server. Επιτρέπει στους κόμβους να έχουν πολλές εικόνες λειτουργικών συστημάτων μεταξύ των πειραμάτων. Αυτή η υπηρεσία δουλεύει πάνω από μια multicast σύνοδο.

   Για τους κόμβους έχουν υλοποιηθεί βιβλιοθήκες βασισμένες στο Linux Kernel. Αυτές είναι:

1. Node Agent. Είναι αντίστοιχο του Node Handler και αποκρίνεται στις εντολές του. Μπορεί να εκκινεί ή να διακόπτει εφαρμογές και να αναφέρει την κατάσταση των πειραμάτων στον ελεγκτή.
2. ORBIT Management Library (OML). Καθορίζει τις δομές δεδομένων και τις συναρτήσεις για αποστολή/λήψη και κωδικοποίηση/αποκωδικοποίηση των δεδομένων των μετρήσεων σε μορφή XDR. Χρησιμοποιείται από τους κόμβους και από τον Collection Server.
3. Libmac. Είναι μια βιβλιοθήκη C που επιτρέπει στις εφαρμογές να παρεμβάλλουν πλαίσια του επιπέδου MAC. Επίσης επιτρέπει τη διαχείριση ασύρματων παραμέτρων όπως ρυθμίσεις του καναλιού TxPower ή θόρυβο συνολικά και ανά πακέτο.

5. Federation

Στο κεφάλαιο αυτό θα γίνει αναφορά στην έννοια του Federation των πόρων μεταξύ των διάφορων πειραματικών πλατφόρμων και θα περιγραφεί το NOVI Project όσο αφορά τη συμβολή του στη κατεύθυνση αυτή.

5.1 Η έννοια του Federation

Η χρήση της εικονικοποίησης από μόνη της με σκοπό τη δημιουργία και την μετέπειτα αξιοποίηση πειραματικών εικονικών υποδομών δεν αρκεί. Χρειάζεται επιπλέον διασύνδεση των επιμέρους εικονικών δικτύων μεταξύ τους με την έννοια της κοινής χρήσης πόρων σε ένα σύνολο διασυνδεδεμένων (federated) υποδομών. Η έννοια του federation εισάγει τη δυνατότητα διασύνδεσης υποδομών που βασίζονται σε διαφορετικού τύπου τεχνικές μεταγωγής για παράδειγμα διασύνδεση μεταξύ υποδομής που βασίζεται στη μεταγωγή πακέτου και υποδομής που βασίζεται στη μεταγωγή κυκλώματος. Η εφαρμογή του feredation επιτρέπει την κοινή χρήση πόρων μεταξύ διάφορων κόμβων πολλών εικονικών περιβαλλόντων που διαφέρουν όσο αφορά τη διαχείριση, την παραμετροποίησή τους, το λογισμικό και το υλικό πάνω στα οποία βασίζονται. Ο στόχος είναι μέσω του federation να ξεπεραστούν αυτά τα εμπόδια.

5.1.1 SFA

Το βασικό πλαίσιο πάνω στο οποίο αναπτύσσεται η δυνατότητα του federation μεταξύ των slices καθώς και ο έλεγχος ορίζεται από ένα κοινό πλαίσιο αναφοράς το SFA (Slice Federation Architecture). Το SFA όπως περιγράφεται από το SFA Draft 1.0 και από το SFA Draft 2.0 επιτρέπει την ενοποίηση διαφορετικών πειραματικών πλατφόρμων μεταξύ τους με την έννοια ότι ένας χρήστης ενός testbed είναι σε θέση να χρησιμοποιήσει πόρους που ανήκουν σε διασυνδεδεμένα με αυτό testbeds. Το SFA διαχωρίζει τους παράγοντες που αλληλεπιδρούν σε ένα federated εικονικό περιβάλλον στους:

- διάφορους ιδιοκτήτες κάθε ενός τμήματος της όλης δικτυακής υποδομής που μπορούν να ορίζουν περιορισμούς για τη χρήση του τμήματος της δικτυακής υποδομής που τους ανήκει,

- διαχειριστές της κάθε δικτυακής υποδομής που εργάζονται προς όφελος των αντίστοιχων ιδιοκτητών της υποδομής με σκοπό την εύρυθμη λειτουργία κάθε υποδομής, την παροχή υπηρεσιών στους ερευνητές και την αντιμετώπιση κακόβουλων ενεργειών,
- ερευνητές που χρησιμοποιούν το σύνολο της δικτυακής υποδομής με απώτερο σκοπό την εκτέλεση των πειραμάτων τους, την υλοποίηση πειραματικών υπηρεσιών,
- Identity anchors που είναι υπεύθυνοι για το authentication των χρηστών.

Στα πλαίσια κάθε SFA αρχιτεκτονικής υφίσταται μια διαχειριστική αρχή (MA – Management Authority) η οποία έχει υπό τον έλεγχό της ένα τμήμα της δικτυακής υποδομής ελέγχοντας κατά πόσο τα στοιχεία που την απαρτίζουν υπακούουν στους εκάστοτε περιορισμούς χρήσης (use policies) καθώς επίσης διενεργώντας την επιθυμητή από τον εκάστοτε ιδιοκτήτη του component δέσμευση και κατανομή πόρων (resource allocation). Κάθε διαχειριστική αρχή διαθέτει μια βάση δεδομένων με τους κόμβους που είναι υπό τον έλεγχό της καθώς και με το που ανήκει ο κάθε κόμβος.
Κάθε SFA διαθέτει επιπλέον μία Αρχή Διαχείρισης για τα slices (SA – Slice Authority) που είναι υπεύθυνη για την ονοματοδοσία του κάθε slice καθώς επίσης και την ενεργοποίηση του δικαιώματος πρόσβασης σε αυτό των χρηστών του. Κάθε SA συνοδεύεται και από ένα contact interface προκειμένου να αντλεί κανείς πληροφορίες για το slice και επίσης να παρακολουθείται η εύρυθμη λειτουργία του.
Ένας χρήστης στην SFA αρχιτεκτονική μπορεί να έχει πολλαπλούς ρόλους. Μπορεί να είναι ένας ερευνητής που επιθυμεί να εκτελέσει ένα πείραμα ή μια υπηρεσία, ένας διαχειριστής (operator) που διαχειρίζεται ένας μέρος της υποδομής, ένας PI από ένα ινστιτούτο που κάνει χρήση της υποδομής για ερευνητικούς σκοπούς, ή ένας ιδιοκτήτης που συνεισφέρει πόρους στην υποδομή.

Components: Στοιχείο (component) στην SFA ορίζεται ένα σύνολο από πόρους, που περιλαμβάνει φυσικούς πόρους (CPU, μνήμη, δίσκος, εύρος ζώνης), λογικούς πόρους αριθμούς θυρών, καθώς και σύνθετους πόρους όπως packet forwarding fast paths. Αυτοί οι πόροι μπορούν να βρίσκονται είτε σε μια συσκευή είτε σε περισσότερες, ανάλογα με τη φύση του component. Τα components ταξινομούνται σε aggregates. Όλα τα components ενός aggregate είναι υπό την διαχείριση της ίδιας MA, που επίσης διαχειρίζεται το aggregate. Κάθε aggregate ελέγχεται από έναν aggregate manager (AM), που διαθέτει για διαχειριστικούς λόγους ένα interface για απομακρυσμένη πρόσβαση. Αν ένα aggregate

διαθέτει μόνο ένα στοιχείο τότε ο AM ονομάζεται διαχειριστής component (CM). Μια διαχειριστική αρχή, υπό τον έλεγχο του ιδιοκτήτη θέτει κανονισμούς για το πώς θα κατανεμηθούν οι πόροι στους διάφορους χρήστες και στα πειράματα τους.

Slices: Μια άλλη βασική οντότητα της SFA αρχιτεκτονικής είναι τα slices. Ένα slice ορίζεται ως ένα δίκτυο υπολογιστικών πόρων που μπορούν να τρέξουν ένα πείραμα.

Οι διαδικασίες που αφορούν την δημιουργία slices χωρίζονται σε τρία βήματα:

- Εγγραφή (Registration) κατά την οποία το slice υφίσταται μόνο σαν όνομα και ορίζονται και οι χρήστες που έχουν πρόσβαση σε αυτό
- Instantiation κατά την οποία εκχωρούνται στο slice συγκεκριμένα στοιχεία της δικτυακής υποδομής καθώς και τμήμα των πόρων που εμπεριέχουν τα στοιχεία αυτά.
- Ενεργοποίηση (Activation) κατά την οποία το slice ενεργοποιείται και δύναται να τρέξει κώδικα εκ μέρους του χρήστη

Κάθε SFA ορίζει global identifiers – GIDs για το σύνολο των αντικειμένων που απαρτίζουν το διασυνδεδεμένο σύστημα όπως, τα components, τα slices, οι υπηρεσίες. Κάθε δηλαδή οντότητα του συστήματος διαθέτει από ένα GID. Χάρις στα GIDs διασφαλίζεται η ασφάλεια ενός συστήματος δεδομένου ότι κάθε οντότητα που διαθέτει ένα GID μπορεί να επιβεβαιώσει ότι αποδόθηκε σύμφωνα με το συγκεκριμένο SFA καθώς και ότι είναι όντως εκείνη που διαθέτει το συγκεκριμένο GID. Πιο συγκεκριμένα το GID είναι ένα πιστοποιητικό που συνοδεύεται από μια τριάδα πληροφοριών, το Public Key, το UUID και το χρόνο ζωής (lifetime). Το αντικείμενο που διαθέτει το GID «απαντά» με το PublicKey. Το UUID – Universally Unique Identifier είναι ένα αναγνωριστικό που είναι ίδιο για τα ίδια αντικείμενα (objects) εντός της δικτυακής υποδομής. Ο χρόνος ζωής (lifetime) προσδιορίζει για πόσο χρόνο είναι έγκυρο ένα GID. Τα GIDs χρειάζεται να ανανεώνονται συχνά.

Η παραπάνω μορφή GID που περιγράφηκε αφορά το απλό GID – Plain GID

Άλλες μορφές GID είναι το signed GID που προκύπτει με προσθήκη μιας signature που παράγεται από εγκεκριμένη αρχή και το bundled GID

Τύποι δεδομένων: Εκτός από τα GIDs η Slice Federation Architecture περιέχει τέσσερις τύπους δεδομένων (RSpec, Ticket, Credentials)

Ο resource specification – RSpec τύπος δεδομένων αναφέρεται στους πόρους ενός δεδομένου component του δικτύου και στους περιορισμούς και στα εμπόδια για την κατανομή τους. Κάθε RSpec περιλαμβάνει τα εξής δύο πεδία:

(StartTime, Duration)

δίνοντας με αυτόν τον τρόπο πληροφορίες για την χρονική διάρκεια για την οποία οι επιθυμητοί πόροι ζητούνται.

Ένας άλλος τύπος δεδομένων είναι το ticket που παράγεται από τον RSpec έπειτα από εντολή της MA ως υπόσχεση στον κάτοχο του ticket για κατανομή πόρων σε μελλοντική χρονική στιγμή.

Ο τύπος δεδομένων έχει την ακόλουθη μορφή

Ticket = (RSpec, GID, SeqNum)

Όπου το RSpec αναφέρεται στους πόρους για τους οποίους έχουν παραχωρηθεί δικαιώματα εκχώρησης, το GID αναφέρεται στην οντότητα που της έχει παραχωρηθεί το δικαίωμα να κατανείμει τους πόρους και το SeqNum εξασφαλίζει την μοναδικότητα του Ticket. Αυτή η πληροφορία υπογράφεται από αυτόν που εξέδωσε το ticket.

Το Credential αφορά στα δικαιώματα και στα προνόμια που έχει κάποιος σε σχέση με μια οντότητα.

Τα προνόμια που αφορούν τα slices είναι τα εξής:

Privileges	Interface	Operations
instantiate	slice	GetTicket, CreateSlice, DeleteSlice, UpdateSlice
Bind	slice	GetTicket, LoanResources
control	slice	UpdateSlice, StopSlice, StartSlice, DeleteSlice
Info	slice	Listslices, ListComponents, GetListResources, GetSliceBySignature
Operator	Management	All

Η χρήση του slice interface εξυπηρετεί τη δημιουργία και τον έλεγχο των slices.

Για το instantiation ενός slice ένας συνδυασμός από 4 διαδικασίες λαμβάνουν χώρα:

Ticket= GetTicket (Credential, RSpec)

RedeemTicket(Ticket)

ReleaseTicket(Ticket)

CreateSlice(Credential, RSpec)

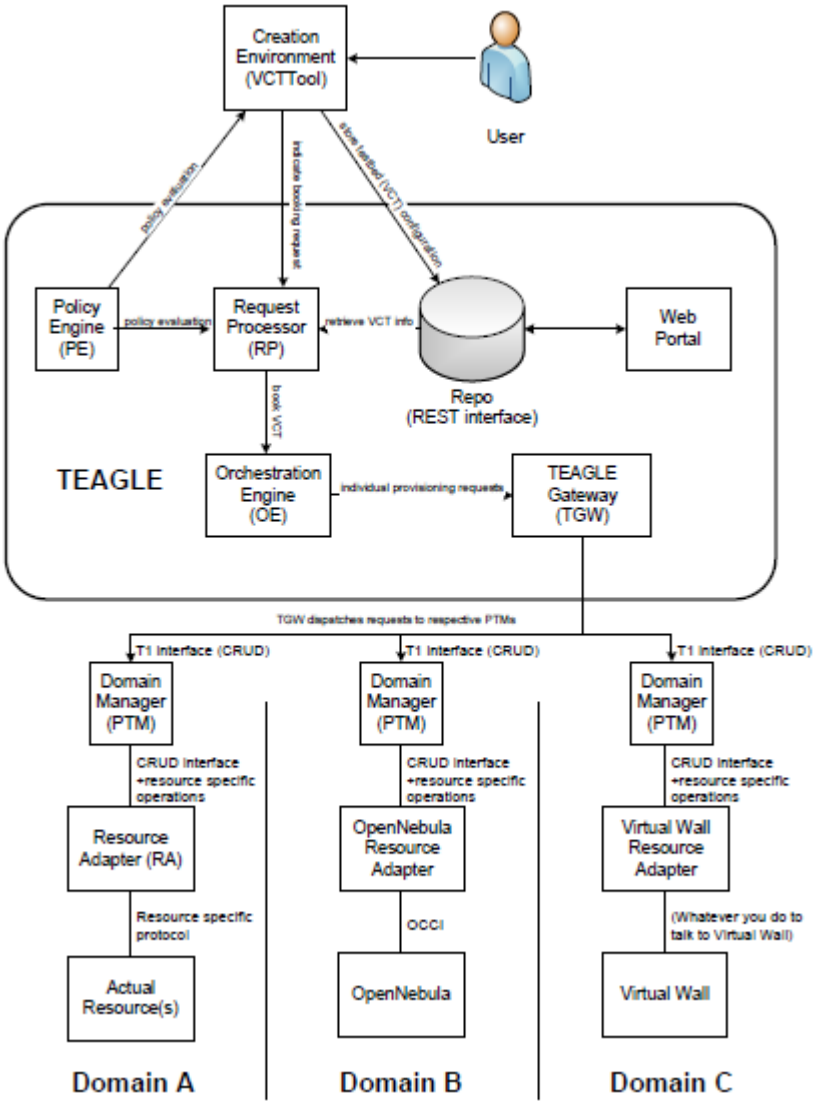
Στα πλαίσια της SFA αρχιτεκτονικής ένας χρήστης που φέρει ένα credential που του δίνει το δικαίωμα να εκτελεί ένα πλήθος λειτουργιών. Πρέπει να σημειωθεί ότι η δημιουργία τοπικών slivers είναι δυνατή μόνο από τον εκάστοτε κόμβο, κατά συνέπεια κατά τη δημιουργία ενός slice η διαδικασία πρέπει να γίνει σε κάθε κόμβο χωριστά.

Βασικές λειτουργίες που μπορούν να εκτελεστούν στα πλαίσια της SFA είναι η προσθήκη ενός χρήστη στην SFA registry, ενός slice και η προσθήκη ενός κόμβου. Οι λειτουργίες

αυτές θα πρέπει να εκτελεστούν από τον administrator. Στη συνέχεια όπως αναφέρθηκε και προηγουμένως η κατανομή των πόρων στο νέο slice γίνεται από την πλευρά του χρήστη του εκάστοτε κόμβου.

5.1.2 Teagle Framework

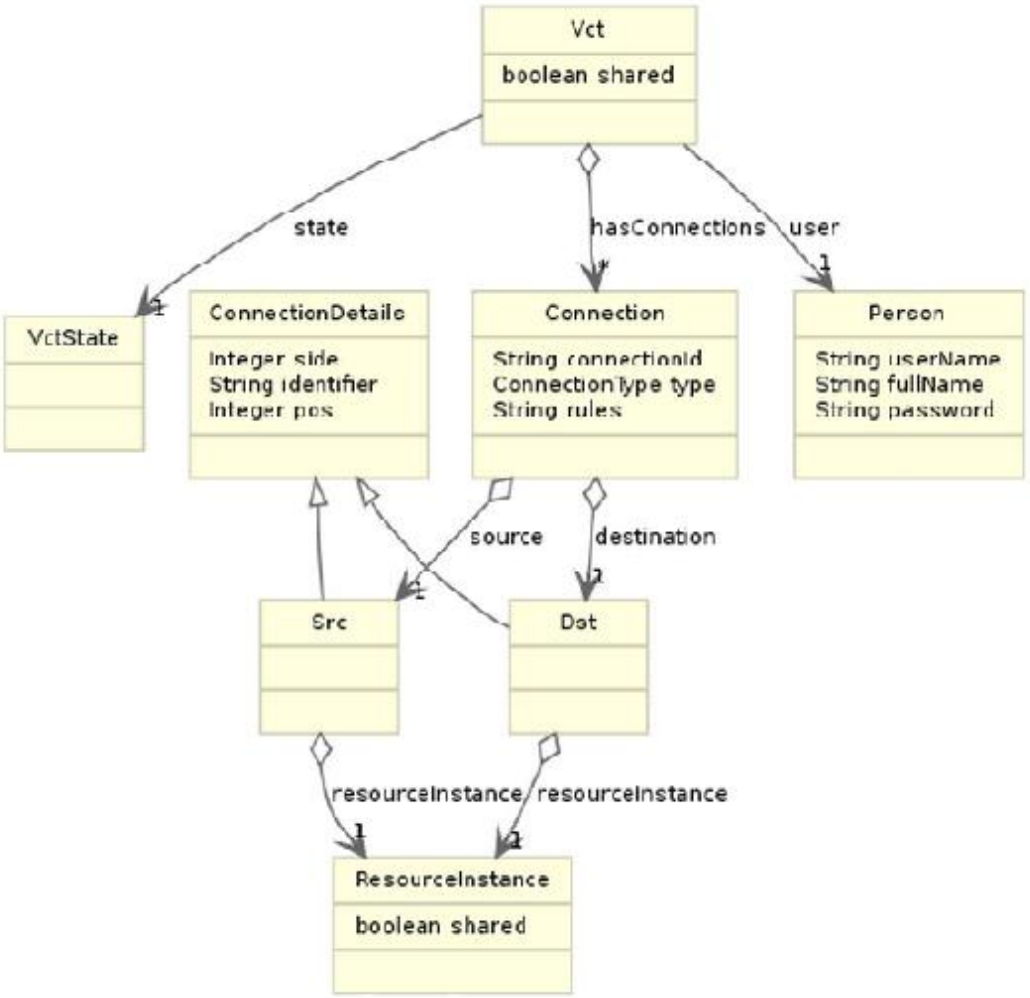
Το Teagle είναι μια αρχιτεκτονική federation των πόρων που έχει αναπτυχθεί στα πλαίσια του PanLab II. Βασικό του χαρακτηριστικό είναι ότι στοχεύει σε μια γενική σχεδίαση που του επιτρέπει την αποτίμηση και διαχείριση κάθε υπηρεσίας.



Κύρια στοιχεία της Teagle αρχιτεκτονικής όπως παρουσιάζονται και στο παραπάνω σχήμα είναι τα εξής:

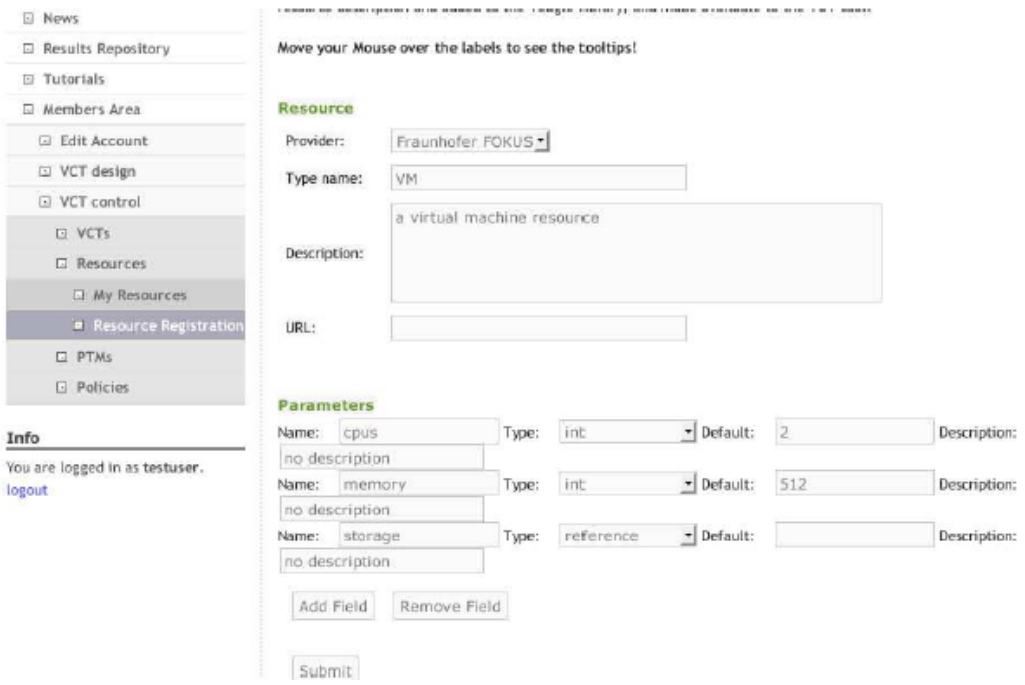
- Ένα web portal
- Ένα model-based repository
- Μια πειραματική πλατφόρμα με κατάλληλες δυνατότητες παραμετροποίησης
- Ένας επεξεργαστής αιτημάτων (request processor)
- Ένας μηχανισμός αστυνόμευσης (policy engine)
- Ένας μηχανισμός ενορχήστρωσης
- Μια πύλη (gateway)

Σκοπός του Teagle είναι η δημιουργία Εικονικών Πειραμάτων Πελάτη (Virtual Customer Testbeds – VCTs). Ένα VCT συνίσταται από διάφορες οντότητες (πόρους) διασυνδεδεμένες μεταξύ τους όπως στο παρακάτω σχήμα:



Μια διάκριση που πρέπει να γίνει αφορά τους resource types και τις resource instances. Ένας resource type με όρους OOP (Object Oriented Programming) πρέπει να νοείται σαν μια κλάση που γενικότερα περιγράφει τη συμπεριφορά μια οντότητας ενώ μια resource instance ως αντικείμενο που μπορεί να έχει τις δικές του τιμές για τα ορίσματα που ο τύπος ορίζει.

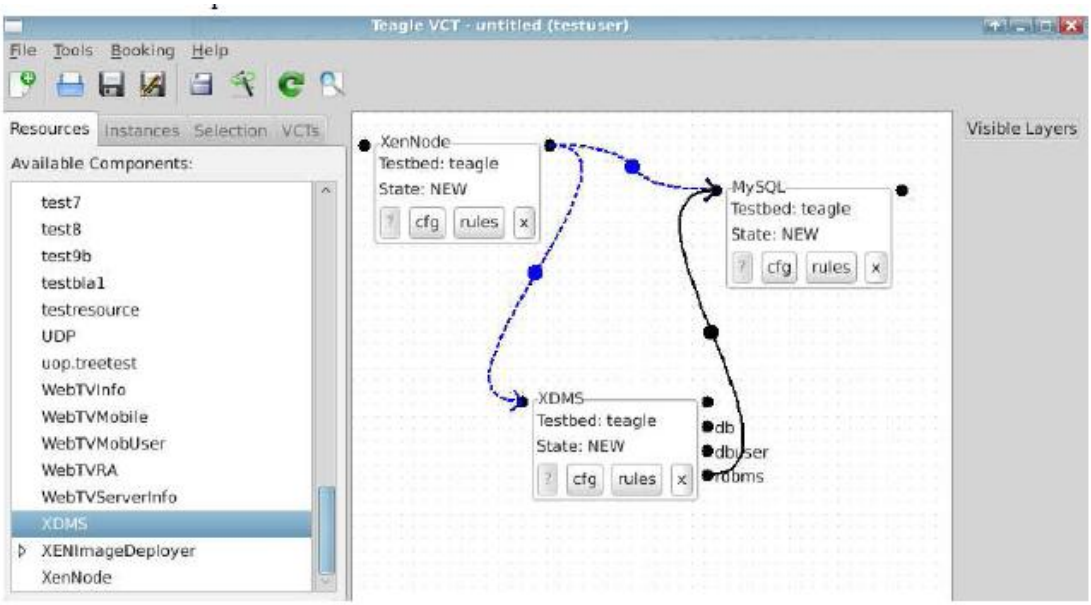
Web portal: Το web portal είναι μια ιστοσελίδα που επιτρέπει στους πελάτες και τους συνεργάτες να χρησιμοποιούν τα εργαλεία του Teagle και τους πόρους του Panlab. Ένας διαχειριστής ενός testbed που επιθυμεί να διαθέσει τους πόρους στο Teagle δεν έχει παρά να εγγράψει τον οργανισμό τους και τους πόρους που επιθυμεί να προσφέρει μέσω των λειτουργιών του portal. Η σελίδα μέσω της οποίας μπορεί να γίνει η περιγραφή των προς διάθεση πόρων είναι η ακόλουθη:



Repository: Το repository είναι ένα στοιχείο κρίσιμης σημασίας της αρχιτεκτονικής του Teagle. Σε αυτό αποθηκεύονται πληροφορίες σχετικά με τους διαθέσιμους resource types, οι παράμετροι από υπάρχουσες resource instances και τα VCTs. Μέσω ιστοσελίδας μπορεί ένας χρήστης να έχει πρόσβαση στα δεδομένα μέσω των HTTP μεθόδων GET, POST, PUT,

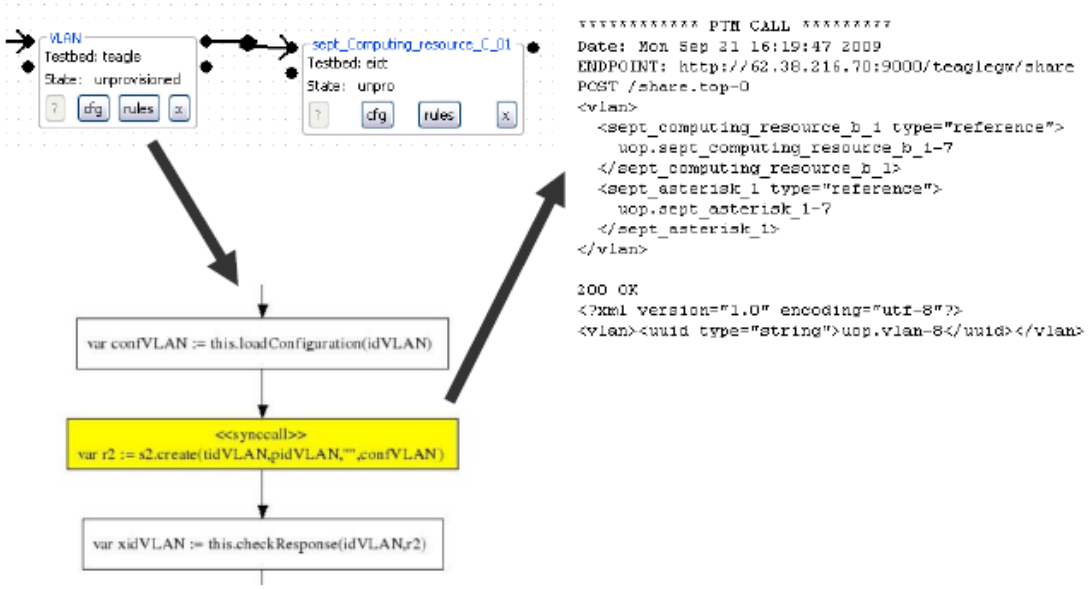
DELETE που αντιστοιχούν σε δημιουργία, ανάγνωση, ανανέωση, και διαγραφή δεδομένων αντίστοιχα.

VCT Tool: Το VCT Tool είναι μια εφαρμογή web όπου οι πελάτες μπορούν να κάνουν κράτηση ενός Εικονικού Πειράματος Πελάτη – Virtual Customer Testbed (VCT). Στο VCT Tool έχουν πρόσβαση εγγεγραμμένοι χρήστες μέσω του Teagle Portal. Βασικός στόχος του VCT Tool είναι οι πελάτες να μπορούν να σχεδιάσουν πειραματικά περιβάλλοντα συνδυάζοντας μια ποικιλία πόρων. Το VCT Tool επικοινωνεί κατά την εκκίνηση του αλλά και κατά τη διάρκεια εκτέλεσης του με το repository με σκοπό τη λήψη δεδομένων που αφορούν τη προσκόμιση ρυθμίσεων, δεδομένων και αποθηκευμένων πειραματικών υποδομών. Η βασική διεπαφή χρήστη του VCT tool περιλαμβάνει τη δυνατότητα σχεδίασης και παραμετροποίησης πόρων στο χώρο εργασίας που παρέχεται. Οι καρτέλες στα αριστερά του panel επιτρέπουν την εξεύρεση των επιθυμητών πόρων, τον έλεγχο της διαθεσιμότητα τους και την εισαγωγή υπαρχόντων πόρων. Στο παρακάτω σχήμα παρουσιάζεται η διεπαφή χρήστη του VCT Tool με ένα απλό testbed.



Αφού έχουν επιλεχθεί οι πόροι που προορίζονται για την δημιουργία ενός VCT και αφού παραμετροποιηθούν σύμφωνα με τις επιθυμίες των πελατών γίνεται κράτηση (booking) του VCT. Το αίτημα κράτησης μαζί με ένα XML αρχείο που θα το περιγράφει αποστέλλεται στον μηχανισμό ενορχήστρωσης (Orchestration Engine) ο οποίος με τη σειρά του είναι υπεύθυνος για την μεταφορά των αιτημάτων για κάθε resource instance ενός VCT στους Domain Managers με τη σωστή σειρά.

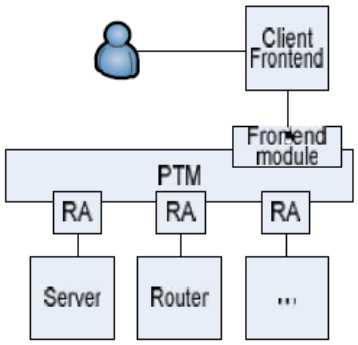
Μηχανισμός ενορχήστρωσης: Ο μηχανισμός ενορχήστρωσης στοχεύει στη δημιουργία ενός VCT πάνω σε ένα ή περισσότερα testbeds που είναι τμήματα μιας federated υποδομής. Ο μηχανισμός ενορχήστρωσης διαθέτει ένα web-based interface ώστε να ορίζει ένα VCT. Το αποτέλεσμα χρήσης αυτού του μηχανισμού είναι μια νέα υπηρεσία η οποία περιγράφεται υπό τη μορφή script.



Το παραπάνω σχήμα περιγράφει την διαδικασία που λαμβάνει χώρα με τη χρήση του μηχανισμού ενορχήστρωσης.

Teagle Gateway: Η πύλη του Teagle διαχειρίζεται service endpoints για όλα τα εγγεγραμμένα PTMs. Τα service endpoints χρησιμοποιούνται κατά τη διάρκεια εκτέλεσης των VCT scripts από τον μηχανισμό ενορχήστρωσης. Προσπελαύνονται ως υπηρεσίες web μέσω 4 διαδικασιών (create, read, update, delete). Η μέθοδος POST χρησιμοποιείται για την δημιουργία και την ανανέωση πόρων, ενώ η GET για τη υποβολή ερωτημάτων (queries).

Domain Managers (PTM): Η διαχειριστική αρχή ενός Teagle domain είναι ένας Panlab Testbed Manager (PTM). Βασική λειτουργία ενός PTM είναι η παροχή γενικών λειτουργιών στους πόρους που είναι υπό τον έλεγχο του. Η δομή ενός PTM domain παρουσιάζεται στο παρακάτω σχήμα:



Λόγω της ετερογένειας που υφίσταται μεταξύ των διάφορων πόρων που διαχειρίζεται ένας PTM δεν μπορεί να είναι σε θέση να γνωρίζει τα semantics όλων των πόρων. Ένα αφαιρετικό στρώμα που αποτελείται από προσαρμογείς πόρων – resource adapters (RA) δημιουργείται. Οι προσαρμογείς αυτοί λειτουργούν ως οδηγοί συσκευών με την έννοια ότι κατέχουν.

Το Teagle επικοινωνεί με τους domain managers μέσω του Teagle gateway διαμέσου της T1 διεπαφής. Οι domain managers είναι υπεύθυνοι για την επεξεργασία αιτημάτων που αφορούν την παροχή πόρων εκ μέρους των domain που διαχειρίζονται.

Η T1 χρησιμοποιείται για αιτήματα σύνδεσης και επικοινωνίας προερχόμενα από οποιαδήποτε πλευρά. Υλοποιείται ως SOAP webservice και διασφαλίζει την ασφάλεια της μέσω SSL/TLS. Το Teagle GW προκαλεί την εκτέλεση μια εκ των εξής υπηρεσιών που προσφέρει το PTM: δημιουργία ενός πόρου, ανανέωση, διαγραφή και ερώτημα που αναφέρεται στην ανάσυρση πληροφοριών σχετικά με την παραμετροποίηση μιας οντότητας.

5.2 NOVI Project

5.2.1 Στόχος και σχεδιαστικές αρχές

Στόχος του NOVI Project είναι η παροχή των αναγκαίων αλγορίθμων, μεθόδων και υπηρεσιών που απαιτούνται για την διαχείριση και τον έλεγχο των slices που προκύπτουν από την ενοποίηση ετερογενών εικονικών πλατφόρμων. Τα απαιτούμενα χαρακτηριστικά που πρέπει να διαθέτει ως μηχανισμός ενοποίησης είναι τα εξής:

- Υποστήριξη τεχνικών και μεθόδων εικονικοποίησης σχετικά με τους εικονικούς πόρους που υφίσταται στις επιμέρους υποδομές και εξαπλώνονται κατά μήκος των διάφορων κόμβων από τους οποίους αυτές συνίσταται.
- Ανεξαρτησία από τον κατασκευαστή του υλικού ή του λογισμικού δεδομένου ότι το υλικό και το λογισμικό που συνιστούν τις εικονικές υποδομές προέρχεται από πληθώρα κατασκευαστών
- Απεικονιστικές μεθόδους μέτρησης της επίδοσης για την ενοποιημένη (federated) εικονική υποδομή. Η περιγραφή πρέπει να γίνεται με ενιαίο τρόπο για τις διάφορες οντότητες εντός του federation.
- Κανονισμούς διαχείρισης ώστε να καθοριστούν περιορισμοί και κανόνες όσο αφορά την πρόσβαση και χρήση στο σύνολο της εικονικοποιημένης υποδομής
- Ο μηχανισμός πρέπει να διαθέτει επιπλέον semantics ώστε να υποστηρίξει context-aware αποφάσεις

5.2.2 Αρχιτεκτονική NOVI

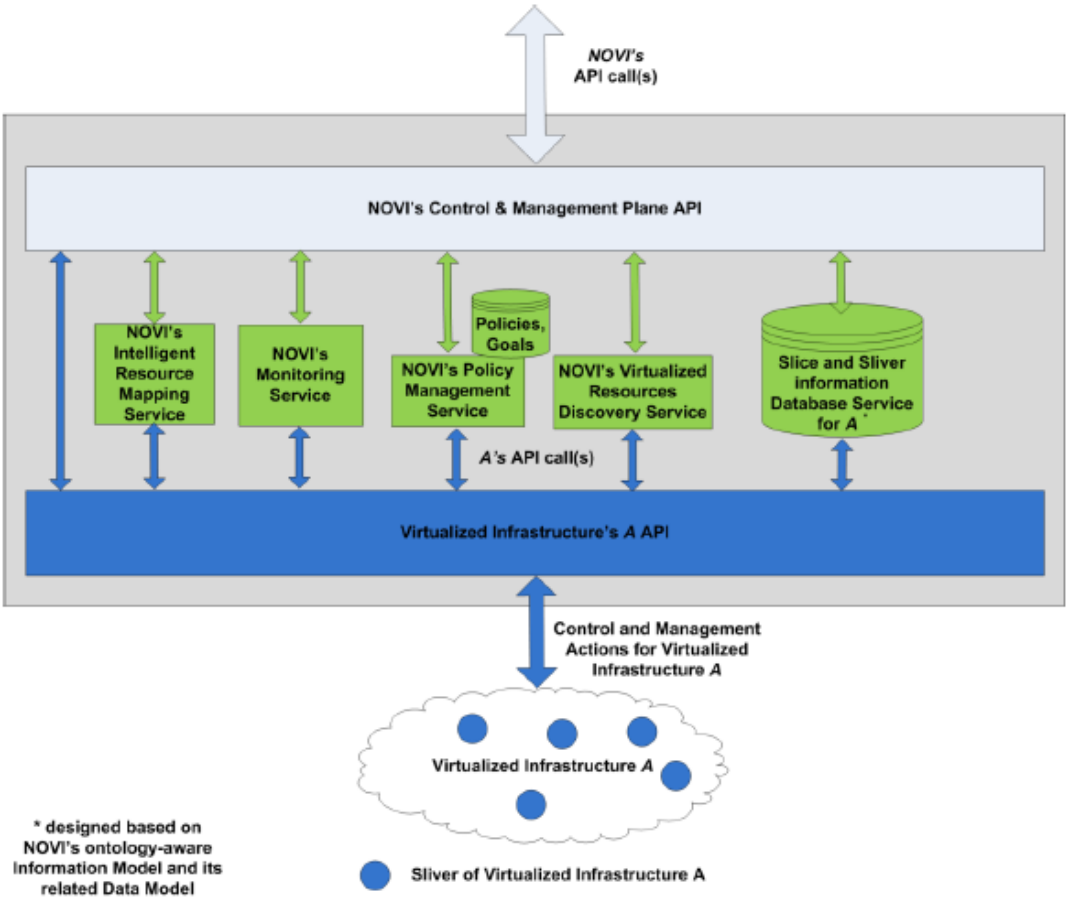
Βασικές οντότητες που περιλαμβάνει το NOVI είναι οι εξής:

- Η υπηρεσία ευφυούς αντιστοίχησης πόρων – Intelligent Resource Mapping Service (IM) που υποστηρίζει την απεικόνιση της εικονικής δικτυακής τοπολογίας στην εκάστοτε federated φυσική δικτυακή υποδομή.
- Η υπηρεσία παρακολούθησης – Monitoring Service που παρέχει τη δυνατότητα στους χρήστες και τους διαχειριστές του NOVI να παρακολουθούν την επίδοση των πόρων σε πραγματικό χρόνο.
- Η υπηρεσία πολιτικών – Policy Managemenent Service που παρέχει λειτουργίες επιβολής συγκεκριμένων περιορισμών όσο αφορά την χρήση των πόρων του εικονικού περιβάλλοντος. Η IM υπηρεσία πρέπει να επιτρέπει το ορισμό το περιορισμών σε αφαιρετικό επίπεδο έτσι ώστε όταν νέες πειραματικές υποδομές συμμετέχουν στο federation να μην χρειάζεται επανακαθορισμός των περιορισμών.
- Μια εικονικοποιημένη διαδικασία ανεύρεσης πόρων – Virtualized Resource Discovery Service (RD) που στόχος της είναι η επιλογή πόρων που είναι κατάλληλοι για την παροχή μιας συγκεκριμένης υπηρεσίας
- Η υπηρεσία βάσης δεδομένων – Database Service διατηρεί πληροφορίες για τα slices και τα slivers των διαχειριζόμενων εικονικών υποδομών. Η βάση δεδομένων

χρησιμοποιείται από όλες τις υπηρεσίες προκειμένου να αποκτήσουν πληροφορίες για τους περιορισμούς που ισχύουν για τους εκάστοτε πόρους.

- Ο request handler που υποστηρίζεται από την IM υπηρεσία και χειρίζεται τα αιτήματα που προέρχονται από χρήστες και πλατφόρμες.

Η αρχιτεκτονική του NOVI αποτυπώνεται στο παρακάτω σχήμα:



5.2.3 Διαδικασία δέσμευσης πόρων

Παρακάτω θα γίνει η περιγραφή ενός αιτήματος για δέσμευση ενός slice σε μια NOVI αρχιτεκτονική που αφορά την ενοποίηση δύο πειραματικών πλατφόρμων. Για τις ανάγκες του παραδείγματος επιλέχθηκε η πλατφόρμα-1 να είναι το PlanetLab και η πλατφόρμα-2 το FEDERICA. Οι διαδικασίες με τη σειρά που λαμβάνουν χώρα είναι οι εξής:

Ένας χρήστης που είναι εγγεγραμμένος χρήστης της πλατφόρμας-1 αιτείται ένα slice για μια εικονική τοπολογία. Τα αιτήματα για εκχώρηση slice, σύμφωνα και με την ορολογία που υιοθετήθηκε και από τους ερευνητές του GENI χωρίζονται σε bound και unbound. Ως bound ορίζονται τα αιτήματα εκείνα κατά τα οποία ο χρήστης ορίζει από ποιους κόμβους και ζεύξεις της φυσικής υποδομής επιθυμεί την δέσμευση των πόρων που αιτείται ενώ

αντίθετα ως unbound αιτήματα ορίζονται εκείνα που δεν ορίζονται οι κόμβοι και οι ζεύξεις από όπου θα δεσμευθούν οι πόροι. Στη συγκεκριμένη use case, υποθέτουμε ότι ο χρήστης αιτείται ένα unbound αίτημα στο NOVI service layer της πλατφόρμας-1. Η αιτούμενη τοπολογία υποστηρίζει πρωτόκολλα δρομολόγησης και παρέχεται από 10 πλήρως meshed routers R1-R10. Η πηγή κίνησης καθώς και ο αποδέκτης της ροής των δεδομένων θα είναι δυο εικονικοί servers S1,S2 διασυνδεδεμένοι με τους εικονικούς δρομολογητές R1 και R10 αντίστοιχα. Επιπλέον, ο χρήστης στο παράδειγμα μας, αιτείται και μια συγκεκριμένη ποιότητα υπηρεσίας όσο αφορά την καθυστέρηση των ζεύξεων πρόσβασης {S1-R1} και {S2-R10}.

Η υπηρεσία αναζήτησης εικονικών πόρων του NOVI θα αναζητήσει στο σύνολο της διαθέσιμης federated υποδομής τους επιθυμητούς πόρους. Οι αιτούμενοι πόροι πρέπει να είναι σύμφωνοι με τις απαιτήσεις που έθεσε ο χρήστης σχετικά με τα λειτουργικά χαρακτηριστικά τους, πχ. Τύποι κόμβων.

Η επιλογή μεταξύ των διαθέσιμων πόρων βελτιστοποιείται μέσω του Intelligent Resource Mapping Service στην πλατφόρμα-1. Κατάλληλοι αλγόριθμοι απεικόνισης εικονικής σε φυσική υποδομή χρησιμοποιούνται ώστε να επιλεχθούν οι πόροι με τον βέλτιστο τρόπο, με βάση και τις διάφορες απαιτήσεις σε ποιότητα υπηρεσίας. Τα αποτελέσματα του αλγορίθμου για το συγκεκριμένο αίτημα οδηγούν στην απεικόνιση των S1 και S2 στους φυσικούς servers S1’ και S2’ της πλατφόρμας-1 και οι δρομολογητές R1-R10 απεικονίζονται στους R1’-R10’ στην πλατφόρμα-2. Τα R2’ και R3’ routers συνδέονται μέσω του εικονικού μεταγωγέα SW1’.

Έχοντας ολοκληρωθεί η εκτέλεση του αλγορίθμου σε αυτό το στάδιο γίνεται η δέσμευση των πόρων και η δημιουργία των slices ταυτόχρονα και στις δύο federated υποδομές. Πιο συγκεκριμένα, το NOVI service layer όσο αφορά την πλατφόρμα-1 ζητά τη δημιουργία των εικονικών εξυπηρετητών S1 και S2 μέσω του προσαρμογέα υποδομής της πλατφόρμας-1 και ταυτόχρονα αιτείται μέσω του προσαρμογέα της πλατφόρμας-2 τη δημιουργία των routers

Πλέον ο χρήστης έχει στη διάθεση του την federated εικονική υποδομή που αιτήθηκε και του επιτρέπει την εκτέλεση πειραμάτων σε αυτή. Είναι σε θέση να παρακολουθεί διάφορες παραμέτρους, κατά μήκος του NOVI slice σε πραγματικό χρόνο, είτε αυτές αφορούν την επίδοση του πειράματος του είτε τη ενδεχόμενη ανάκυψη ενός προβλήματος σχετικά με την τοπολογία που αιτήθηκε.

Οι δεσμευμένοι πόροι καθώς και η δυνατότητα παρακολούθησης τους λήγουν μετά από ένα συγκεκριμένο χρονικό διάστημα. Ο χρήστης μπορεί να παρατείνει τη δέσμευση πόρων που

έχει κάνει ή να αιτηθεί τη διαγραφή του slice πριν τη λήξη του. Διαφορετικά, οι διαδικασίες παρακολούθησης του slice διακόπτονται και οι δεσμευμένοι πόροι απελευθερώνονται.

Με βάση το παραπάνω παράδειγμα που αφορά στην αλληλουχία βημάτων που λαμβάνουν χώρα κατά τη δέσμευση ενός slice αναλύονται οι αλληλεπιδράσεις μεταξύ των διάφορων στοιχείων του NOVI σε όλα τα στάδια δημιουργίας ενός slice είτε κατά το στάδιο του αιτήματος και της δημιουργίας νέου slice.

1. Όταν ένα αίτημα έρχεται στο NOVI service layer μέσω του NOVI request handler, αφού το αίτημα γίνει αποδεκτό από την υπηρεσία πολιτικών (Policy Manager Service - PolMan),
2. η υπηρεσία ευφυούς απεικόνισης πόρων (IRMap) ενεργοποιεί την κατανεμημένη διαδικασία ανεύρεσης πόρων μέσω της Εικονικοποιημένης διαδικασίας ανεύρεσης πόρων – Virtualized Recourse Discovery Service (RD) κατά μήκος των federated πλατφόρμων. Στη παραπάνω Use case η υπηρεσία RD ελέγχει τη διαθεσιμότητα συγκεκριμένων πόρων στους φυσικούς κόμβους όπως CPU, μνήμη RAM και χωρητικότητα στους κόμβους και καθυστέρηση για κάθε ζεύξη μεταξύ των εξυπηρετητών.
3. Για αυτό το λόγο η RD υπηρεσία απευθύνεται στην υπηρεσία παρακολούθησης του δικτύου – Monitoring Service (MonSrv) και αναμένει απάντηση σχετικά με τη διαθεσιμότητα και θέση των αναγκαίων προς υλοποίηση της τοπολογίας πόρων. Σε αυτό το στάδιο το IRMap διαθέτει όλες τις αναγκαίες πληροφορίες για να επιλέξει ανάμεσα στους διαθέσιμους πόρους.
4. Στη συνέχεια το IRMap επιτυγχάνει μέσω των προσαρμογέων υποδομής – Infrastructure Adapters (IAs) τη δέσμευση των απαιτούμενων πόρων με κατανεμημένο τρόπο. Οι προσαρμογείς υποδομής είναι ακόμα υπεύθυνοι και για την εγκατάσταση λογισμικού που είναι απαραίτητο για την υπηρεσία απεικόνισης των μετρήσεων. Αφού δημιουργηθεί η εικονική υποδομή εγγράφεται στη βάση δεδομένων που χρησιμοποιείται για την αποθήκευση στοιχείων σχετικά με τα slices και τα slivers.
5. Στη συνέχεια το IRMap ενημερώνει το MonSrv για να αρχίσει την απεικόνιση μιας λίστας ιδιοτήτων και είτε να στείλει σήμα για συγκεκριμένα γεγονότα στον PolMan.
6. Τελικά ο PolMan ενημερώνει τον χρήστη ότι το slice που αιτήθηκε είναι έτοιμο για χρήση.
7. Σε περίπτωση που η υπηρεσία απεικόνισης (MonSrv) εντοπίσει ότι ένα στοιχείο του slice αντιμετωπίζει πρόβλημα ή υπάρξει κάτι που υπερβαίνει τα όρια που έχει θέσει ο

χρήστης για μια παράμετρο π.χ η καθυστέρηση ξεπεράσει ένα αποδεκτό όριο, τότε η υπηρεσίας αστυνόμευσης ενημερώνει τον χρήστη και κάνει τις ανάλογες ενέργειες. Μετά από ένα τέτοιο συμβάν το slice «επαναπεικονίζεται» στους διαθέσιμους πόρους.

Βιβλιογραφία

[1]N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba, Network Virtualization: State of the Art and Research Challenges

[2]Wendell Odom, CCENT/CCNA ICND1 Official Exam Certification Guide, Cisco Press

[3]www.vmware.com

[4]www.xen.org

[5]http://en.wikipedia.org/wiki/Memory_virtualization

[6]N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba, A survey of Network Virtualization

[7]Xuxian Jiang and Dongyan Xu, VIOLIN: Virtual Internetworking on Overlay Infrastructure

[8] Cloud Computing Hai Jin, Shadi Ibrahim, Tim Bell, Li Qi, Haijun Cao, Song Wu, and Xuanhua Shi

[9]Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, Mike Wawrzoniak, Operating System Support for Planetary-Scale Network Services

[10] Jorge Carapinha, Javier Jiminez, Network Virtualization – A View from the bottom

[11] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network Architecture", ACM Computer Communication Review, vol. 26, no. 2, 1996.

[12] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research", IEEE Communications Magazine, vol. 35, no. 1, pp. 80-86, January 1997.

[13] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks", SIGCOMM Computer Communication Review, vol. 29, no. 2, pp. 7-23,1999.

[14] Salim Hariri, Weiming Wang, Subhra Saha, Vijay Radhakrishnan, Kartik Dev. A. J, Seungchan Oh, Kyou Ho Lee, Gil Young Choi, "Survey & Classification of Programmable Network Technologies"

[15] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh, "Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols"

[16] Stephen Soltesz, Herbert P¨otzl_, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson, “Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors”

[17] http://www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf , December 31, 2009

[18] System virtualization profile, DMTF profile DSP1042, ver. 1.0.0, http://www.dmtf.org/sites/default/files/standards/documents/DSP1042_1.0.0_0.pdf

[19] Linux V-Server, http://linux-vserver.org/Welcome_to_Linux-VServer.org

[20] OpenVZ, http://wiki.openvz.org/Main_Page

[21] VirtualBox, http://www.virtualbox.org/wiki/VirtualBox

[22] Click, http://www.read.cs.ucla.edu/click/

[23] Mark Handley Orion Hodson Eddie Kohler. “XORP: An Open Platform for Network Research,” ACM SIGCOMM Hot Topics in Networking, 2002.

[24] GENI System Overview, http://www.geni.net/

[25] UML, http://user-mode-linux.sourceforge.net/

[26] Mauro Campanella, “The FEDERICA Project: creating cloud infrastructures”

[27] Jeroen van der Ham, Chrysa Papagianni, Jozsef Steger, Peter Matray, Yiannos Kryftis, Paola Grosso, Leonidas Lymperopoulos, “Challenges of an Information Model for Federating Virtualized Infrastructures”

[28] Gregor Schaffrath, Anja Felmann, Andreas Wundsam, Christoph Werle, Ronald Bless, Mario Kind, Laurent Mathy, Panagiotis Papadimitriou, Adam Greenhalgh, Olaf Maennel “Network Virtualization Architecture: Proposal and Initial Prototype”