ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΥΣΤΗΜΑΤΑ P2P – ΣΥΣΤΗΜΑΤΑ ΝΕΦΟΥΣ

Διδακτικές Σημειώσεις για το μάθημα

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Διδάσκων: Κωνσταντίνος Αντωνής
Δρ. Μηχανικός Η/Υ & Πληροφορικής Πανεπιστημίου Πατρών

Λαμία, Οκτώβριος 2018

Ευχαριστίες

Για το κεφάλαιο 1 χρησιμοποιήθηκε ένα μέρος της πτυχιακής εργασίας της φοιτήτριας μου στο ΤΕΙ Λαμίας Σύλβιας Μαραβέλα, την οποία και ευχαριστώ πολύ για την πολύ καλή εργασία της.

Περιεχόμενα

1. Peer-to-peer Συστήματα 10

1.1 Εισαγωγή 10
1.2 Ορισμός του Peer-to-peer Υπολογισμού 11
1.3 Peer – to – Peer και Grid Υπολογισμός 12
1.4 Κατηγορίες των Peer-to-peer Εφαρμογών 13
Επικοινωνία και συνεργασία (Communication and Collaboration) 13
Κατανεμημένος υπολογισμός (Distribution Computation) 13
Υποστήριξη υπηρεσιών Διαδικτύου (Internet Service Support) 14
Συστήματα βάσεων δεδομένων (Database Systems) 14
Διαμοίραση περιεχομένου (Content Distribution) 14
1.5 Peer-to-Peer Content Distribution Συστήματα 14
Peer-to-peer Εφαρμογές (Peer-to-Peer Applications) 15
Peer-to-Peer Υποδομές (Peer-to-Peer Infrastructures) 15
1.6 Τοποθέτηση αντικειμένων και δρομολόγηση 18
Κεντρικοποίηση Overlay Δικτύου 19
Δομή Δικτύου (Network Structure) 20
1.7 Μη δομημένες αρχιτεκτονικές (Unstructured Architectures) 22
Υβριδικές Αποκεντρικοποιημένες (Hybrid Decentralized) 22
Πλήρως αποκεντρικοποιημένα συστήματα 24
Μερικώς Κεντρικοποιημένα (Partially Centralized) 26
1.8 Δομημένες Αρχιτεκτονικές 27
Freenet–Ένα χαλαρά δομημένο σύστημα 28
Chord 31
CAN 33
Tapestry 35

1.9 Distributed Hash Tables 38
Chord 41
2. Υπολογισμός σε Νέφος (Cloud Computing) 47
2.1 Τεχνολογίες 49
Εικονικοποίηση (Virtualization) 49
Αρχιτεκτονική προσανατολισμένη στην υπηρεσία (service – oriented architecture) 51
Υπολογισμός σε πλέγμα (Grid Computing) 52
Utility computing 53
2.2 Υποδομές 53
Χαρακτηριστικά Υποδομών 54
2.3 Μοντέλα Υπηρεσιών 56
Υποδομή σαν Υπηρεσία (Infrastructure as a Service – IaaS) 56
Πλατφόρμα σαν Υπηρεσία (Platform as a Service) 58
Λογισμικό σαν Υπηρεσία (Software as a Service – SaaS) 59
Η εξέλιξη του Λογισμικού σαν Υπηρεσία 60
Το μοντέλο εξέλιξης των τεσσάρων βημάτων 60
Ταυτότητα σαν Υπηρεσία (Identity as a Service – IDaaS) 61
Δίκτυο σαν Υπηρεσία (Network as a Service) 63
2.4 Διαχείριση 64
2.5 Αποθήκευση Δεδομένων 65
2.6 Εικονικοποίηση 70
2.7 Ασφάλεια 77
2.8 Κινητός Υπολογισμός σε Νέφος 79
2.9 Αρχιτεκτονική Κατανεμημένων Εφαρμογών 80
Επίπεδο Χρήστη 81
Επίπεδο Εφαρμογής 81
Επίπεδο πρόσβασης στα Δεδομένα 81

Επίπεδο Αποθήκευσης Δεδομένων 81
2.10 Αρχιτεκτονική Λογισμικού Σαν Υπηρεσία (SaaS Architecture) 82
Επίπεδο Κατανομής (Distribution tier) 82
Άλλα συστατικά επιπέδου εφαρμογής 83
Εξυπηρετητής διαχείρισης ταυτοτήτων 83
Εξυπηρετητής ενοποίησης 83
Εξυπηρετητής επικοινωνιών 83
3. Βιβλιογραφικές Αναφορές 84

Κατάλογος Πινάκων

Πίνακας 1. Κατηγοριοποίηση των Peer-to-Peer Content Distribution συστημάτων με βάση τη δομή του δικτύου και παραδείγματα αυτών. 21

Πίνακας 2. Ο χάρτης που υπάρχει στον κόμβο με ID 67493..... 36

Πίνακας 3. Ο καθορισμός των μεταβλητών για τον κόμβο n, χρησιμοποιώντας m bits identifiers. 44

Κατάλογος Εικόνων

Εικόνα 1: Ο τρόπος που διάφορα χαρακτηριστικά σχεδίασης επηρεάζουν τα κύρια χαρακτηριστικά των peer-to-peer content distribution συστημάτων. 18

Εικόνα 2: Μια αρχιτεκτονική ενός αποκεντρικοποιημένου υβριδικού peer-to-peer συστήματος. Ένας κεντρικός υπολογιστής καταλόγου (central directory server) διατηρεί έναν δείκτη των metadata για όλους τα αρχεία του δικτύου. 23

Εικόνα 3: Ένα παράδειγμα του Gnutella μηχανισμού αναζήτησης. 26

Εικόνα 4: Η χρήση των έμμεσων αρχείων στο Freenet. 31

Εικόνα 5: Ένας Chord "κύκλος ταυτοτήτων" που αποτελείται από τρεις κόμβους 0,1 και 3. Το κλειδί 1 τοποθετείται στον κόμβο 1, το κλειδί 2 στον 3 και το κλειδί 6 στον 0. 32

Εικόνα 6: (a) Ένα παράδειγμα με 2-d [0,1] x [0,1] χώρο χωρισμένο μεταξύ 5 CAN κόμβων (b) Ένα παράδειγμα 2-d χώρου μετά από την σύνδεση του κόμβου F. 34

Εικόνα 7: Tapestry: Ένα παράδειγμα μηχανισμού δρομολόγησης Plaxton mess με δεκαδικά ψηφία μήκους 5. 37

Εικόνα 8: (a). Η συχνότητα κατανομής του DHT των κόμβων αποθηκεύοντας έναν σταθερό αριθμό documents, (b) Τον αριθμό των κόμβων που δεν έχουν documents, (c) Ο μικρότερος, ο μέσος όρος και ο μεγαλύτερος αριθμός των documents ανά κόμβο. 41

Εικόνα 9: Ένας identifier circle που περιέχει τρεις κόμβους 0,1 και 3. 43

Εικόνα 10: (a) Τα finger διαστήματα του κόμβου 1. (b) Το finger table και το κλειδί για ένα δίκτυο με κόμβους 0,1 και 3 και κλειδιά 1,2 και 6. 45

Εικόνα 11: Εικονικοποίηση. 51

Εικόνα 12: Αρχιτεκτονική προσανατολισμένη στην υπηρεσία. 52

Εικόνα 13: Υπολογισμός σε πλέγμα. 53

Εικόνα 14: Συστατικά υποδομών νέφους. 54

Εικόνα 15: Χαρακτηριστικά υποδομών νέφους. 55

Εικόνα 16: Υποδομή σαν υπηρεσία. 56

Εικόνα 17: Πλατφόρμα σαν υπηρεσία. 58

Εικόνα 18: Single Sign-On. 62

Εικόνα 19: Υλοποίηση Single Sign-On. 63

Εικόνα 20: Δίκτυο σαν Υπηρεσία για κινητά δίκτυα. 64

Εικόνα 21: Η αρχιτεκτονική ενός γενικού παράλληλου συστήματος αρχείων. 67
Εικόνα 22: Η αρχιτεκτονική του Google File System (GFS). 68
Εικόνα 23: Η αρχιτεκτονική του Hadoop. 69
Εικόνα 24: Σύστημα αποθήκευσης νέφους. 70
Εικόνα 25: Hypervisor (τύπος 1). 73
Εικόνα 26: Hypervisor (τύπος 2). 74
Εικόνα 27: Hypervisor (τύπος 3). 74
Εικόνα 28: Πλήρης εικονικοποίηση. 75
Εικόνα 29: Εξομοίωση. 76
Εικόνα 30: Paravirtualization 77
Εικόνα 31: Η αρχιτεκτονική του Xen. 77
Εικόνα 32: Cloud Security Aliance model. 79
Εικόνα 33: Αρχιτεκτονική κινητού υπολογισμού σε νέφος. 80

1. Peer-to-peer Συστήματα

1.1 Εισαγωγή

Η χρήση των υπολογιστών διανύει μια χρονική περίοδο κατά την οποία βρίσκεται σε εξέλιξη μια επανάσταση. Οι περισσότεροι οργανισμοί διέθεταν μόνο ένα μικρό αριθμό υπολογιστών που, λόγω της ανυπαρξίας ενός τρόπου για να διασυνδεθούν, λειτουργούσαν συνήθως ανεξάρτητα.

Η κατάσταση αυτή άρχισε να διαφοροποιείται με την ανάπτυξη ισχυρών μικροεπεξεργαστών και την δημιουργία τοπικών δικτύων ή LAN υψηλών ταχυτήτων. Το τελικό αποτέλεσμα των δυο αυτών τεχνολογιών είναι ότι πλέον δεν είναι μόνο δυνατό, αλλά και εύκολο, να λειτουργήσουν μαζί υπολογιστικά συστήματα που αποτελούνται από ένα μεγάλο αριθμό από CPU που συνδέονται μέσω δικτύων υψηλών ταχυτήτων. Τα συστήματα αυτά αποκαλούνται κατανεμημένα συστήματα.

Η βάση της λειτουργίας των κατανεμημένων συστημάτων, όπως τα Gnutella, Seti@Home, OceanStore, κλπ., είναι η “peer-to-peer”, ένα νέο κύμα στις αρχιτεκτονικές δικτύων. Αυτές οι αρχιτεκτονικές χαρακτηρίζονται γενικά από την άμεση διανομή των υπολογιστικών πόρων (αποθηκευτικοί χώροι, περιεχόμενα) χωρίς την μεσολάβηση του κεντρικού υπολογιστή.

Το κίνητρο για την χρήση των peer-to-peer εφαρμογών προέρχεται σε μεγάλο βαθμό από τη δυνατότητά τους να λειτουργούν, να κλιμακώνονται και να οργανώνονται παρουσία ενός μεγάλου πληθυσμού κόμβων (nodes), δικτύου και αποτυχιών υπολογιστών χωρίς την ανάγκη ενός κεντρικού υπολογιστή. Τέτοιες αρχιτεκτονικές έχουν ως χαρακτηριστικά την ικανότητα κλιμάκωσης (scalability), αντίσταση στη λογοκρισία, τον κεντρικοποιημένο έλεγχο και την αυξανόμενη πρόσβαση στους πόρους. Η διαχείριση, η συντήρηση, η ευθύνη για τη λειτουργία, ακόμη και η έννοια “της ιδιοκτησίας” των peer-to-peer συστημάτων διανέμεται μεταξύ των χρηστών, αντί του χειρισμού από μια ενιαία επιχείρηση ή κάποιο πρόσωπο. Τέλος, οι peer-to-peer αρχιτεκτονικές έχουν τη δυνατότητα να επιταχύνουν τις διαδικασίες επικοινωνίας και να μειώνουν το κόστος συνεργασίας μέσω της τυχαίας (ad-hoc) διαχείρισης των ομάδων εργασίας.

Στο κείμενο αυτό ερευνούμε την τεχνολογία διαμοίρασης περιεχομένων (content distribution) peer-to-peer με τη χρήση των Κατανεμημένων Πινάκων Κατακερματισμού (Distributed Hash Tables - DHTs), η οποία στοχεύει να παρέχει μια περιεκτική περιγραφή των εφαρμογών, των χαρακτηριστικών γνωρισμάτων και των τεχνικών εφαρμογής. Δεδομένου ότι αυτός είναι ένας νέος και γρήγορα εξελισσόμενος τομέας, καθώς νέες δυνατότητες εισάγονται συνεχώς, αυτό το κείμενο θα παρουσιάσει την παρούσα κατάσταση την τρέχουσα χρονική στιγμή - όπως γίνεται αναπόφευκτα σε κάθε

περίπτωση για οποιαδήποτε έρευνα που αφορά έναν ερευνητικό τομέα που αναπτύσσεται. Ωστόσο, πιστεύουμε ότι οι πληροφορίες που παρουσιάζονται θα παραμείνουν σχετικές και χρήσιμες για τον αναγνώστη.

1.2 Ορισμός του Peer-to-peer Υπολογισμού

Μια γρήγορη επισκόπηση της βιβλιογραφίας αποκαλύπτει έναν σημαντικό αριθμό διαφορετικών ορισμών των “peer-to-peer” συστημάτων, που διακρίνονται κυρίως από τη "ευρύτητα" που προσάπτουν στον όρο.

Οι ακριβέστεροι ορισμοί για τα συστήματα peer-to-peer αναφέρονται σε πλήρως κατανεμημένα συστήματα, στα οποία όλοι οι κόμβοι είναι απολύτως ισοδύναμοι από την άποψη της λειτουργίας και των ενεργειών που εκτελούν. Αυτοί οι ορισμοί αδυνατούν να καλύψουν τα συστήματα που υιοθετούν, παραδείγματος χάριν, την έννοια των “supernodes” (κόμβοι που λειτουργούν ως τοπικοί-κεντρικοί υπολογιστές) όπως το Kazaa, τα οποία είναι, ευρέως αποδεκτά.

Σύμφωνα με έναν ευρύτερο και γενικώς αποδεκτό ορισμό [S2000], το “peer-to-peer” είναι μια κατηγορία εφαρμογών που εκμεταλλεύονται τους πόρους (αποθήκευση, κύκλοι ΚΜΕ, περιεχόμενο, ανθρώπινη παρουσία), που είναι διαθέσιμοι στις άκρες του διαδικτύου. Αυτός ο ορισμός, ωστόσο, καλύπτει και τα συστήματα που στηρίζονται στους κεντρικοποιημένους servers για τη λειτουργία τους, όπως το Seti@home, καθώς επίσης και στις διάφορες εφαρμογές από τον τομέα του Grid computing.

Γενικά, μπορεί να ειπωθεί ότι δεν υπάρχει καμία γενική συμφωνία για αυτό που “είναι” και για αυτό που “δεν είναι” peer-to-peer.

Θεωρούμε ότι αυτή η έλλειψη συμφωνίας για έναν ορισμό οφείλεται, σε μεγάλο βαθμό, στο γεγονός ότι τα συστήματα ή οι εφαρμογές ονομάζονται “peer-to-peer” όχι λόγω της εσωτερικής λειτουργίας ή της αρχιτεκτονικής τους, αλλά λόγω του πώς γίνονται αντιληπτά “εξωτερικά”, δηλαδή εάν δίνουν την εντύπωση της παροχής της άμεσης αλληλεπίδρασης μεταξύ των υπολογιστών. Κατά συνέπεια, οι διαφορετικοί ορισμοί “peer-to-peer” εφαρμόζονται για να προσαρμόσουν τις διαφορετικές περιπτώσεις τέτοιων συστημάτων ή εφαρμογών. Τα δύο στοιχεία που καθορίζουν τα χαρακτηριστικά των peer-to-peer αρχιτεκτονικών είναι τα ακόλουθα:

1. Ο *διαμοιρασμός των υπολογιστικών πόρων* μέσω απ’ ευθείας ανταλλαγής χωρίς την μεσολάβηση ενός κεντρικοποιημένου server. Οι κεντρικοποιημένοι servers μπορούν μερικές φορές να χρησιμοποιηθούν για συγκεκριμένες ενέργειες, όπως για παράδειγμα για την προσθήκη νέων κόμβων (nodes) στο δίκτυο ή για την απόκτηση

σφαιρικών κλειδιών για την κρυπτογράφηση των δεδομένων. Παρ' όλα αυτά, τα συστήματα που στηρίζονται σε έναν ή περισσότερους καθολικούς κεντρικοποιημένους servers για βασικές λειτουργίες τους, όπως η αποθήκευση και η αναζήτηση μέσω σφαιρικών δεικτών (π.χ. Napster, Publius) επεκτείνουν σαφώς τον ορισμό των peer-to-peer συστημάτων. Καθώς οι κόμβοι του peer-to-peer δικτύου δεν μπορούν να στηρίζονται σ' ένα κεντρικό υπολογιστή που θα συντονίζει την ανταλλαγή περιεχομένου και τη λειτουργία ολόκληρου του δικτύου, πρέπει να συμμετέχουν ενεργά μέσω ανεξάρτητων και μονομερών ενεργειών. Παραδείγματα τέτοιων ενεργειών είναι η έρευνα για άλλους κόμβους, ο εντοπισμός ή η ενδιάμεση αποθήκευση των δεδομένων, η δρομολόγηση πληροφορίας και μηνύματος, η σύνδεση ή η αποσύνδεση από άλλους γειτονικούς κόμβους, η κρυπτογράφηση, η εισαγωγή, η ανάκτηση, η αποκρυπτογράφηση και ο έλεγχος του περιεχομένου, κ.α.

2. Η *δυνατότητά τους να προσαρμόζονται αυτόματα* στις αποτυχίες των συνδέσεων των δικτύων και των υπολογιστών, καθώς επίσης και σε έναν παροδικό πληθυσμό κόμβων. Αυτή η ανοχή σε σφάλματα (fault-tolerant) και η ικανότητα να οργανώνονται από μόνοι τους (self-organizing) υποδηλώνουν την ανάγκη για μια προσαρμοστική τοπολογία δικτύων η οποία θα αλλάξει, καθώς οι κόμβοι εισάγονται ή φεύγουν και οι συνδέσεις των δικτύων αποτυγχάνουν ή ανακτώνται, προκειμένου να διατηρηθεί η συνδεσιμότητα και η απόδοσή τους.

Λαμβάνοντας υπόψη τα παραπάνω στοιχεία προκύπτει ο ακόλουθος ορισμός:

Τα peer-to-peer συστήματα είναι κατανεμημένα συστήματα που αποτελούνται από αλληλοσυνδεδεμένους κόμβους ικανούς να προσαρμόζονται αυτόματα (self-organizing) στην τοπολογία του δικτύου με σκοπό τη διανομή των πόρων, όπως το περιεχόμενο, οι κύκλοι της ΚΜΕ, ο αποθηκευτικός χώρος και το εύρος ζώνης, καθώς και να είναι ικανοί στις αποτυχίες και την παροχή φιλοξενίας στους παροδικούς κόμβους διατηρώντας την συνδεσιμότητα και την απόδοση, χωρίς την μεσολάβηση ενός κεντρικού υπολογιστή. [AS2004]

Αυτός ο ορισμός καλύπτει "τους βαθμούς κεντρικοποιήσης" οι οποίοι κυμαίνονται από τα αμιγή συστήματα, τα οποία είναι εντελώς αποκεντρικοποιημένα συστήματα, όπως το Gnutella, ως τα "μερικώς κεντρικοποιημένα" όπως το Kazaa. Ωστόσο, η παρουσίαση των αρχιτεκτονικών και των συστημάτων δεν θα περιοριστεί στον ορισμό αυτό, αλλά θα ληφθούν υπόψη και τα συστήματα peer-to-peer που χρησιμοποιούν έναν κεντρικό υπολογιστή όπως το Napster.

1.3 Peer – to – Peer και Grid Υπολογισμός

Τα peer-to-peer καθώς και τα grid συστήματα είναι δύο παραλλαγές κατανεμημένου υπολογισμού, που και οι δύο ενδιαφέρονται για την οργάνωση της διαμοίρασης των πόρων σε μεγάλης κλίμακας υπολογιστικά συστήματα.

Τα grids είναι κατανεμημένα συστήματα που επιτρέπουν τη μεγάλης κλίμακας συντονισμένη χρήση και το διαμοιρασμό γεωγραφικά κατανεμημένων πόρων, βασισμένα σε μόνιμες, προτυποποιημένες υποδομές εξυπηρέτησης, έχοντας συχνά σαν σκοπό την υψηλή απόδοση.

Καθώς τα συστήματα τύπου grid μεγαλώνουν κλιμακωτά, απαιτούν λύσεις σε θέματα όπως η αυτοδιαμόρφωση (self-configuration), η αντοχή σε σφάλματα και η ικανότητα κλιμάκωσης, όπου η έρευνα στα peer-to-peer συστήματα έχει πολλά να προσφέρει.

Συμπερασματικά μπορούμε να πούμε ότι τα grid συστήματα είναι προσανατολισμένα κυρίως προς τις υποδομές, χωρίς να ενδιαφέρονται τόσο για την αντοχή σε σφάλματα, ενώ τα peer-to-peer συστήματα ενδιαφέρονται λιγότερο για τις υποδομές και περισσότερο για την αντοχή σε σφάλματα. Επιπλέον, η μορφή της διαμοίρασης όπως αυτή αντιμετωπίστηκε από τα peer-to-peer συστήματα είχε περιορισμένη λειτουργικότητα, παρέχοντας καθολική διαμοίραση περιεχομένου και χώρο διαμοίρασης αρχείων, χωρίς όμως οποιαδήποτε μορφή ελέγχου πρόσβασης.

Καθώς όμως οι peer-to-peer τεχνολογίες χρησιμοποιούνται σε περισσότερο «έξυπνες» και πολύπλοκες εφαρμογές, όπως η δομημένη διαμοίραση περιεχομένου, η συνεργασία σε επίπεδο επιφάνειας εργασίας, οι υπολογισμοί πάνω από δίκτυο, είναι φανερό ότι θα υπάρξει σύγκλιση μεταξύ των δύο αυτών τεχνολογιών.

1.4 Κατηγορίες των Peer-to-peer Εφαρμογών

Οι peer-to-peer αρχιτεκτονικές έχουν υιοθετηθεί για πολλές και διαφορετικές κατηγορίες εφαρμογών, οι οποίες αναλύονται παρακάτω:

Επικοινωνία και συνεργασία (Communication and Collaboration)

Αυτή η κατηγορία περιλαμβάνει τα συστήματα που παρέχουν την υποδομή για επικοινωνία και συνεργασία μεταξύ των peer υπολογιστών σε πραγματικό χρόνο. Παραδείγματα τέτοιων εφαρμογών είναι οι εφαρμογές ανταλλαγής μηνυμάτων σε πραγματικό χρόνο και το chat.

Κατανεμημένος υπολογισμός (Distribution Computation)

Αυτή η κατηγορία περιλαμβάνει τα συστήματα που έχουν στόχο να εκμεταλλευθούν την υπολογιστική ισχύ των nodes (CPU κύκλοι). Αυτό επιτυγχάνεται με το χωρισμό ενός task σε μικρές μονάδες εργασίας και τη διανομή τους σε διαφορετικούς peers, οι οποίοι εκτελούν την αντίστοιχη εργασία τους και επιστρέφουν τα αποτελέσματα. Ο κεντρικός συντονισμός απαιτείται κυρίως για το χωρισμό και τη διανομή των tasks καθώς και για την συλλογή των αποτελεσμάτων. Παραδείγματα τέτοιων συστημάτων περιλαμβάνουν τα προγράμματα όπως το Seti@home.

Υποστήριξη υπηρεσιών Διαδικτύου (Internet Service Support)

Ένας μεγάλος αριθμός εφαρμογών βασισμένος στις peer-to-peer υποδομές έχει προκύψει για την υποστήριξη ποικίλων υπηρεσιών Διαδικτύου. Παραδείγματα τέτοιων εφαρμογών είναι τα peer-to-peer multicast συστήματα και οι εφαρμογές ασφάλειας (security applications), που παρέχουν την προστασία ενάντια στην άρνηση εξυπηρέτησης (denial of service) ή στις επιθέσεις των ιών.

Συστήματα βάσεων δεδομένων (Database Systems)

Σημαντική δουλειά έχει γίνει στο σχεδιασμό των βάσεων δεδομένων των κατανεμημένων συστημάτων βασισμένων στις peer-to-peer υποδομές. Παραδείγματα τέτοιων βάσεων δεδομένων είναι το Local Relational Mode (LRM), το PIER και το Edutella. Το **LRM** [BGKMSZ2002] είναι ένα μοντέλο σύμφωνα με το οποίο το σύνολο των αποθηκευμένων δεδομένων ενός peer-to-peer δικτύου θεωρείται ότι αποτελείται από ασύμβατες τοπικές σχεσιακές βάσεις δεδομένων που διασυνδέονται από τα σύνολα "γνωριμιών" που καθορίζουν τους κανόνες μετάφρασης των πληροφοριών μεταξύ των συστημάτων αυτών, καθώς και τις σημασιολογικές εξαρτήσεις μεταξύ τους. Το **PIER** [HHLL2003] είναι μια κλιμακούμενη κατανεμημένη μηχανή αναζήτησης ερωτήσεων που στηρίζεται σε μια peer-to-peer υπερκείμενη (overlay) τοπολογία δικτύου η οποία επιτρέπει σε σχεσιακές ερωτήσεις (relational queries) να εκτελεστούν σε χιλιάδες υπολογιστές. Τέλος, το σύστημα **Piazza** [HIMT2003] παρέχει την υποδομή για την δημιουργία των σημασιολογικών εφαρμογών Ιστού (semantic Web applications), που αποτελούνται από κόμβους που μπορούν να παρέχουν είτε πηγαία δεδομένα (π.χ. από μια σχεσιακή βάση δεδομένων), τα σχήματα (schemas) είτε και τα δύο.

Διαμοίραση περιεχομένου (Content Distribution)

Τα περισσότερα από τα σύγχρονα peer-to-peer συστήματα εμπίπτουν στην κατηγορία των συστημάτων διαμοίρασης περιεχομένου, η οποία περιλαμβάνει συστήματα και υποδομές που σχεδιάζονται για το διαμοιρασμό των δεδομένων μεταξύ των χρηστών. Τα peer-to-peer συστήματα διαμοίρασης περιεχομένου κυμαίνονται από τις σχετικά απλές εφαρμογές άμεσης διαμοίρασης αρχείων (direct filesharing) ως τα συστήματα που δημιουργούν ένα κατανεμημένο μέσο αποθήκευσης για ασφαλή και αποτελεσματική δημοσίευση (publishing), οργάνωση, δεικτοδότηση (indexing), εύρεση, ενημέρωση και ανάκτηση των δεδομένων. Μερικά παραδείγματα είναι: το Gnutella [G2003] και το Kazaa [Kazaa, 2003].

1.5 Peer-to-Peer Content Distribution Συστήματα

Ένα peer-to-peer content distribution σύστημα δημιουργεί ένα κατανεμημένο μέσο αποθήκευσης που επιτρέπει την δημοσίευση, την αναζήτηση και την ανάκτηση των αρχείων από τα μέλη του δικτύου του. Δεδομένου ότι τα συστήματα γίνονται περιπλοκότερα, τα μη λειτουργικά χαρακτηριστικά μπορούν να εξασφαλιστούν,

συμπεριλαμβανομένων των παροχών για την ασφάλεια, την ανωνυμία, τη δικαιοσύνη (fairness), την ικανότητα κλιμάκωσης (scalability) και την απόδοση (performance), καθώς επίσης και τη διαχείριση και οργάνωση των πόρων.

Οι σύγχρονες peer-to-peer τεχνολογίες ομαδοποιούνται ως εξής:

Peer-to-peer Εφαρμογές (Peer-to-Peer Applications)

Η κατηγορία αυτή περιλαμβάνει τα content distribution συστήματα που είναι βασισμένα στην peer-to-peer τεχνολογία και διαιρείται στις εξής δύο κατηγορίες [AS2004], που βασίζονται στους στόχους εφαρμογής και την αντιληπτή πολυπλοκότητά τους:

- ***Peer-to-peer "file exchange" systems***

Αυτά τα συστήματα έχουν ως στόχο την ανταλλαγή των αρχείων (files) μεταξύ των peers. Χρησιμοποιούνται για την οργάνωση ενός δικτύου από peers και παρέχουν υπηρεσίες για αναζήτηση (searching) και για μεταφορά των αρχείων μεταξύ τους. Αυτές είναι χαρακτηριστικά απλές εφαρμογές που υιοθετούν μια καλύτερη προσέγγιση χωρίς να ασχολούνται με την ασφάλεια, τη διαθεσιμότητα (availability) και τη μονιμότητα (persistence) της πληροφορίας. Είναι κυρίως συστήματα που είναι αρμόδια για την καταγραφή των ενεργειών ενός χρήστη σε ένα peer-to-peer δίκτυο (reputation).

- ***Peer-to-peer content publishing and storage systems***

Τα συστήματα αυτά έχουν ως στόχο να δημιουργήσουν ένα αποθηκευτικό χώρο μέσα στο οποίο οι χρήστες θα είναι σε θέση να δημοσιεύουν, να αποθηκεύουν και να κατανείμουν τα δεδομένα με ασφαλή και μόνιμο τρόπο. Τα δεδομένα αυτά είναι προσπελάσιμα με έναν ελεγχόμενο τρόπο από τους peers που έχουν τα κατάλληλα δικαιώματα. Τα συστήματα αυτά εστιάζονται στην ασφάλεια και τη μονιμότητα της πληροφορίας και συχνά ο στόχος είναι να ενσωματωθούν λειτουργίες για την υπευθυνότητα, την ανωνυμία και την αντίσταση στον αυστηρό έλεγχο, καθώς επίσης και τις μόνιμες υπηρεσίες διαχείρισης περιεχομένου (ενημέρωση, διαγραφή, έλεγχος έκδοσης).

Peer-to-Peer Υποδομές (Peer-to-Peer Infrastructures)

Αυτή η κατηγορία περιλαμβάνει τις peer-to-peer υποδομές, οι οποίες δεν αποτελούν εφαρμογές, αλλά παρέχουν τις peer-to-peer θεμελιώδεις υπηρεσίες και τα πλαίσια εφαρμογής. Οι θεμελιώδεις αυτές υπηρεσίες υποδομών είναι οι εξής:

- ***Δρομολόγηση και Θέση (Routing and location)***

Ένα peer-to-peer content distribution σύστημα βασίζεται σε ένα δίκτυο από peers μέσα στο οποίο οι αιτήσεις (requests) και τα μηνύματα (messages) πρέπει να δρομολογούνται με αποδοτικότητα (efficiency) και ανεκτικότητα σε σφάλματα (fault tolerance). Με αυτόν τον τρόπο η εύρεση των peers και των δεδομένων γίνεται αποδοτική.

Διαφορετικές υποδομές και αλγόριθμοι έχουν αναπτυχθεί για να παρέχουν τέτοιες υπηρεσίες.

- ***Ανωνυμία (Anonymity)***

Peer-to-peer θεμελιώδη συστήματα έχουν σχεδιαστεί με αποκλειστικό στόχο την ανωνυμία των χρηστών.

- ***Reputation Management***

Σε ένα peer-to-peer δίκτυο, δεν υπάρχει κεντρική οργάνωση για να καταγράψει τις ενέργειες των χρηστών και τη συμπεριφορά τους. Αυτές οι πληροφορίες, επομένως, φιλοξενούνται στους διάφορους κόμβους (nodes) των δικτύων. Με σκοπό να κρατηθούν αυτές οι πληροφορίες ασφαλείς, επίκαιρες και διαθέσιμες σε όλο το δίκτυο πρέπει να αναπτυχθούν σύνθετες reputation management υποδομές.

Τα μη λειτουργικά χαρακτηριστικά των peer-to-peer κατανεμημένων συστημάτων είναι τα ακόλουθα [SG1995]:

- ***Ασφάλεια (Security)***

Υπάρχουν διάφορες όψεις της ασφάλειας :

  - ***Ακεραιότητα και αυθεντικότητα (Integrity and authenticity)***

Διατήρηση των δεδομένων στη μορφή που πρέπει να έχουν. Μη εξουσιοδοτημένοι χρήστες δεν μπορούν να τα τροποποιούν.

  - ***Μυστικότητα και εμπιστευτικότητα (Privacy and confidentiality)***

Αφορά τον τρόπο προστασίας των δεδομένων. Εξασφαλίζει ότι τα δεδομένα θα είναι προσιτά μόνο σε εκείνους που έχουν δικαίωμα πρόσβασης και ότι υπάρχει έλεγχος για το ποια δεδομένα είναι, πώς χρησιμοποιούνται και πώς διατηρούνται.

  - ***Διαθεσιμότητα και μονιμότητα (Availability and persistence)***

Εξασφαλίζει στους εξουσιοδοτημένους χρήστες να έχουν πρόσβαση στα δεδομένα όταν το επιθυμούν. Για τα peer-to-peer content distribution συστήματα αυτό γίνεται πάντα. Αυτό έχει ως αποτέλεσμα να εξασφαλίζεται σταθερότητα είτε στην περίπτωση αποτυχίας (failure) είτε στην περίπτωση αλλαγής των πληθυσμών των κόμβων.

- ***Ικανότητα Κλιμάκωσης (Scalability)***

Η διατήρηση της απόδοσης του συστήματος δεν εξαρτάται από τον αριθμό των κόμβων ή των εγγράφων (documents) στο δίκτυο. Μια αύξηση του αριθμού των κόμβων ή των εγγράφων θα έχει ελάχιστη επίδραση στην απόδοση και τη διαθεσιμότητα.

- ***Απόδοση (Performance)***

Ο χρόνος που απαιτείται για την εκτέλεση των διαδικασιών που επιτρέπονται από το σύστημα. Παράδειγμα τέτοιων διαδικασιών είναι η δημοσίευση, η αναζήτηση (searching) και η ανάκτηση των εγγράφων.

- ***Δικαιοσύνη (Fairness)***

Εξασφαλίζει στους χρήστες έναν δίκαιο και ισορροπημένο τρόπο κατανάλωσης των πόρων του δικτύου.

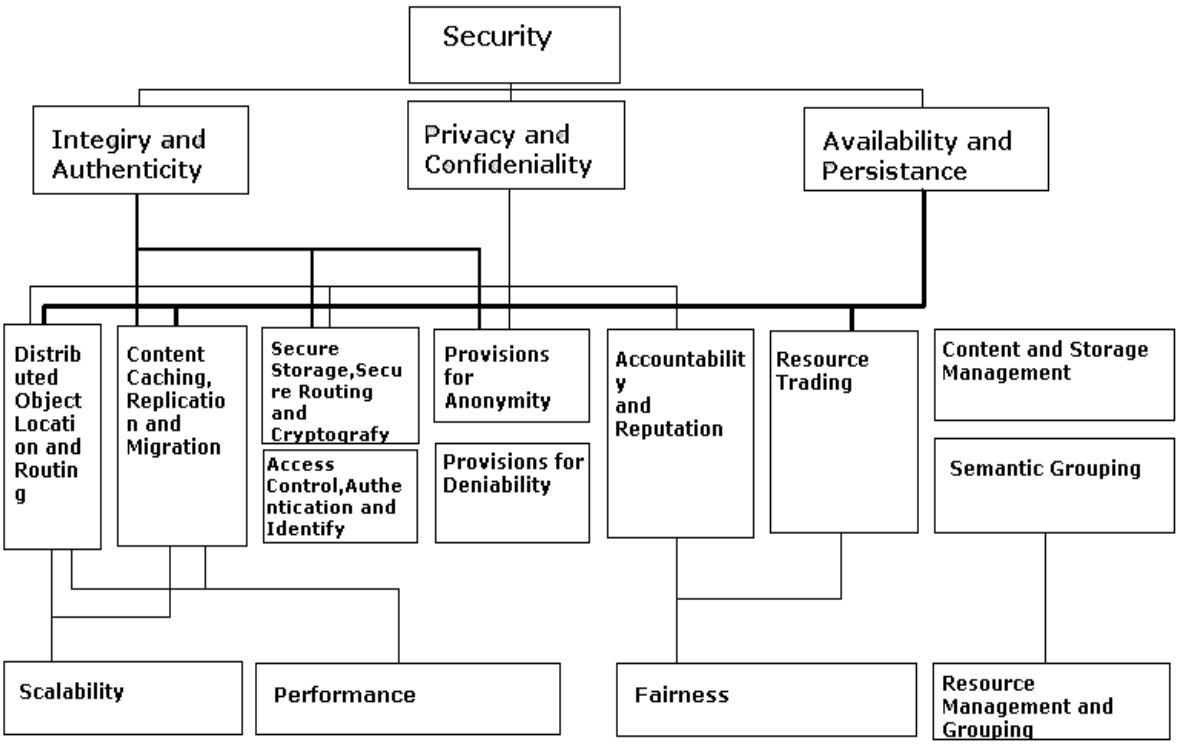
- ***Ικανότητα διαχείρισης πόρων (Resource Management Capabilities)***

Τα peer-to-peer κατανεμημένα συστήματα επιτρέπουν την δημοσίευση, την αναζήτηση και την ανάκτηση των δεδομένων. Περιπλοκότερα συστήματα παρέχουν πιο προηγμένες διαχειριστικές ικανότητες πόρων όπως η επιμέλεια (editing) ή η διαγραφή (removal) των εγγράφων, η διαχείριση του αποθηκευτικού χώρου και οι λειτουργίες πάνω σε μεταδεδομένα (metadata).

- ***Σημασιολογική ομαδοποίηση των πληροφοριών (Semantic Grouping of Information)***

Υπάρχουν διάφορα σχήματα ομαδοποίησης όπως η σημασιολογική ομαδοποίηση που είναι βασισμένη στο ίδιο το περιεχόμενο, ομαδοποίηση που είναι βασισμένη στη θέση ή στην απόσταση των δικτύων, κ.α.

Οι σχέσεις μεταξύ των μη λειτουργικών χαρακτηριστικών απεικονίζονται στην Εικόνα 1 σε ένα UML διάγραμμα, καθώς επίσης και οι σχέσεις τους με τα διάφορα χαρακτηριστικά σχεδιασμού των peer-to-peer κατανεμημένων συστημάτων.



Εικόνα 1: Ο τρόπος που διάφορα χαρακτηριστικά σχεδίασης επηρεάζουν τα κύρια χαρακτηριστικά των peer-to-peer content distribution συστημάτων.

Τα ορθογώνια παραλληλόγραμμα στο κέντρο παρουσιάζουν τις αποφάσεις σχεδιασμού που έχουν επιπτώσεις σε αυτές τις ιδιότητες. Παραδείγματος χάριν παρατηρείται ότι η απόδοση ενός peer-to-peer κατανεμημένου συστήματος επηρεάζεται από τους κατανεμημένους μηχανισμούς τοποθέτησης και δρομολόγησης αντικειμένου (Distributed Object Location and Routing mechanisms), καθώς επίσης και από τους αλγορίθμους αντιγραφής δεδομένων (data replication), ενδιάμεσης αποθήκευσης δεδομένων (Cashing) και μετακίνησης δεδομένων (Migration). Η δικαιοσύνη, από την άλλη μεριά, εξαρτάται από τις παροχές του συστήματος όσον αφορά την υπευθυνότητα και τη φήμη που συνοδεύει τους χρήστες μέσω καταγραφής των ενεργειών τους (Accountability and Reputation), καθώς επίσης και από οποιουσδήποτε μηχανισμούς διακίνησης πόρων (Resource Trading) που μπορούν να εφαρμοστούν.

1.6 Τοποθέτηση αντικειμένων και δρομολόγηση

Η λειτουργία οποιουδήποτε peer-to-peer κατανεμημένου συστήματος στηρίζεται σε ένα δίκτυο από peer υπολογιστές (nodes) και τις συνδέσεις τους (edges). Αυτό το δίκτυο

διαμορφώνεται στην κορυφή ενός βασικού φυσικού δικτύου υπολογιστών – και ανεξάρτητα από αυτό- και αναφέρεται ως “υπερκείμενο (overlay)” δίκτυο. Η τοπολογία, η δομή και ο βαθμός κεντρικοποίησης του overlay δικτύου καθώς και η δρομολόγηση και ο μηχανισμός τοποθέτησης για τα μηνύματα και το περιεχόμενο είναι σημαντικά για τη λειτουργία του συστήματος, καθώς αυτά έχουν επιπτώσεις στην ανεκτικότητα σε σφάλματα, στην αυτό-συντήρηση, στην προσαρμοστικότητα σε αποτυχία, στην απόδοση, στην ικανότητα κλιμάκωσης και στην ασφάλεια. Τα overlay δίκτυα μπορούν να διαιρεθούν με κριτήρια το βαθμό κεντρικοποίησης και τη δομή τους.

Κεντρικοποίηση Overlay Δικτύου

Αν και τα αμιγή overlay peer-to-peer δίκτυα θεωρούνται ολοκληρωτικά αποκεντρικοποιημένα, στην πραγματικότητα αυτό δεν ισχύει πάντα και υπάρχουν συστήματα με διάφορους βαθμούς κεντρικοποίησης. Συγκεκριμένα, αναγνωρίζονται τρεις κατηγορίες:

- ***Αμιγώς αποκεντρικοποιημένες αρχιτεκτονικές (Purely Decentralized Architectures)***

Όλοι οι κόμβοι στο δίκτυο εκτελούν ακριβώς τα ίδια tasks, ενεργώντας τόσο ως εξυπηρετητές (servers) και ως πελάτες (clients) χωρίς την ύπαρξη κεντρικού συντονισμού για τις ενέργειες τους. Οι κόμβοι τέτοιων δικτύων καλούνται "*servents*" (SERVers+clieENTS).

- ***Μερικώς κεντρικοποιημένες αρχιτεκτονικές (Partially Centralized Architectures)***

Η βάση είναι η ίδια με τα καθαρώς αποκεντρικοποιημένα συστήματα. Μερικοί από τους κόμβους, ωστόσο, αναλαμβάνουν έναν σημαντικότερο ρόλο, αυτό των τοπικών κεντρικών κόμβων δεικτοδότησης για τα διαμοιραζόμενα αρχεία των τοπικών peers. Οι κόμβοι τέτοιων δικτύων καλούνται “υπερκόμβοι (*supernodes)*”. Ο τρόπος με τον οποίο ορίζεται ο ρόλος των *supernodes* από το δίκτυο ποικίλλει μεταξύ των διαφορετικών συστημάτων. Είναι σημαντικό, όμως, να σημειωθεί ότι αυτοί οι κόμβοι ορίζονται δυναμικά και σε περίπτωση αποτυχίας το δίκτυο θα λάβει αυτόματα μέτρα να αντικατασταθούν με άλλους.

- ***Αποκεντρικοποιημένες υβριδικές αρχιτεκτονικές (Hybrid Decentralized Architectures)***

Σε αυτά τα συστήματα υπάρχει ένας κεντρικός υπολογιστής που διευκολύνει την σύνδεση μεταξύ των peers διατηρώντας καταλόγους μεταδεδομένων που περιγράφουν τα διαμοιραζόμενα αρχεία που αποθηκεύονται από τους κόμβους. Αν και η end to end διασύνδεση και ανταλλαγή αρχείου μπορούν να πραγματοποιηθούν άμεσα μεταξύ δύο κόμβων, οι κεντρικοί υπολογιστές διευκολύνουν αυτήν την διαδικασία εκτελώντας την αναζήτηση και την ταυτοποίηση των κόμβων που αποθηκεύουν τα αρχεία. Προφανώς σε

αυτές τις αρχιτεκτονικές υπάρχει ένα μοναδικό σημείο αποτυχίας, ο κεντρικός υπολογιστής. Αυτό τις καθιστά μη εύκολα κλιμακούμενες, ευπαθείς στον αυστηρό έλεγχο, σε μια τεχνική αποτυχία ή σε μια κακόβουλη επίθεση.

Δομή Δικτύου (Network Structure)

Με την έννοια δομή αναφερόμαστε στο εάν το overlay δίκτυο δημιουργείται με τρόπο μη ντετερμινιστικό, δηλαδή όπως οι κόμβοι και το περιεχόμενο προστίθενται τυχαία, ή εάν η δημιουργία της δομής είναι βασισμένη σε συγκεκριμένους κανόνες. Από άποψη δομής τα peer-to-peer δίκτυα ταξινομούνται ως εξής [AS2004]:

- ***Μη δομημένα peer-to-peer δίκτυα (Unstructured)***

Η τοποθέτηση του περιεχομένου (αρχεία) είναι απολύτως ανεξάρτητη από την overlay τοπολογία. Αυτό σημαίνει ότι τα αρχεία είναι διάσπαρτα αποθηκευμένα.

Σε ένα μη δομημένο δίκτυο, το περιεχόμενο πρέπει να τοποθετηθεί. Μηχανισμοί αναζήτησης κυμαίνονται από τις μεθόδους brute force όπως η μαζική υπερφόρτωση (flooding) του δικτύου με τη μετάδοση ερωτήσεων με έναν αλγόριθμο ψαξίματος πρώτα σε πλάτος (breadth-first) ή ψαξίματος πρώτα σε βάθος (depth-first) μέχρι το επιθυμητό περιεχόμενο να τοποθετηθεί, μέχρι και περιπλοκότερες στρατηγικές που περιλαμβάνουν τη χρήση των τυχαίων βημάτων (random walks) και των δεικτών δρομολόγησης (routing indices). Οι μηχανισμοί αναζήτησης που χρησιμοποιούνται στα μη δομημένα δίκτυα έχουν επιπτώσεις, ιδιαίτερα στα θέματα διαθεσιμότητας, ικανότητας κλιμάκωσης και μονιμότητας των δεδομένων.

Τα μη δομημένα συστήματα είναι γενικά πιο κατάλληλα για την παροχή φιλοξενίας κόμβων που χαρακτηρίζονται κυρίως ως προσωρινοί. Μερικά αντιπροσωπευτικά παραδείγματα των μη δομημένων συστημάτων είναι το Napster, το Gnutella, το Kazaa, το Edutella, κ.ά.

- ***Δομημένα peer-to-peer δίκτυα (Structured)***

Έχουν προκύψει κυρίως από μία προσπάθεια να αντιμετωπίσουν τα ζητήματα της ικανότητας κλιμάκωσης με τα οποία τα μη δομημένα συστήματα έχουν προβλήματα. Στα δομημένα δίκτυα η overlay τοπολογία ελέγχεται στενά και τα αρχεία (ή δείκτες τους) αποθηκεύονται σε συγκεκριμένες θέσεις. Αυτά τα συστήματα παρέχουν ουσιαστικά μια αντιστοίχηση (mapping) μεταξύ του περιεχομένου και της θέσης (π.χ. διεύθυνση κόμβων), υπό μορφή κατανεμημένου πίνακα δρομολόγησης, έτσι ώστε οι ερωτήσεις (queries) να μπορούν να δρομολογούνται αποτελεσματικά στον κόμβο με το επιθυμητό περιεχόμενο.

Τα δομημένα συστήματα προσφέρουν μια κλιμακούμενη λύση για τις exact-match ερωτήσεις, δηλαδή τις ερωτήσεις στις οποίες ο ακριβής ταυτοποιητής (identifier) των ζητούμενων στοιχείων είναι γνωστός (σε σύγκριση με τις ερωτήσεις λέξης-κλειδιά). Η

χρησιμοποίηση των exact-match ερωτήσεων ως υπόστρωμα για τις ερωτήσεις λέξης-κλειδιά παραμένει ένα ανοικτό ερευνητικό πρόβλημα για τα κατανεμημένα περιβάλλοντα [WMB1999].

Ένα μειονέκτημα των δομημένων συστημάτων είναι ότι είναι δύσκολο να διατηρηθεί η δομή που απαιτείται για την αποδοτική δρομολόγηση των μηνυμάτων, λόγω του ότι κόμβοι συνδέονται και αποσυνδέονται με υψηλούς ρυθμούς [LRS2002]. Χαρακτηριστικά παραδείγματα των δομημένων συστημάτων είναι το Chord, το CAN, το PAST καθώς και άλλα συστήματα.

Μια κατηγορία δικτύων που είναι μεταξύ των δομημένων και των μη δομημένων αναφέρεται ως χαλαρά δομημένα (loosely structured) δίκτυα. Αν και η θέση του περιεχομένου δεν είναι πλήρως καθορισμένη, επηρεάζεται από τις υπόνοιες δρομολόγησης (routing hints). Ένα χαρακτηριστικό παράδειγμα είναι το Freenet [CSW 2000, CHSW 2002].

Ο Πίνακας 1 συνοψίζει τις κατηγορίες που περιγράψαμε, με τα παραδείγματα των peer-to-peer κατανεμημένων συστημάτων και των αρχιτεκτονικών τους. Σημειώστε ότι τα δομημένα και τα χαλαρά δομημένα συστήματα είναι καθαρώς αποκεντρικοποιημένα.

Πίνακας 1. Κατηγοριοποίηση των Peer-to-Peer Content Distribution συστημάτων με βάση τη δομή του δικτύου και παραδείγματα αυτών.

	Centralization		
	Hybrid	**Partial**	**None**
Unstructured	Napster	Kazaa, Gnutella, Edutella	Gnutella
Structured Infrastructures			Chord, CAN
Structured Systems			PAST

Στις επόμενες ενότητες αναλύεται η τοπολογία του overlay δικτύου και η λειτουργία των διαφορετικών peer-to-peer συστημάτων σύμφωνα με το βαθμό κεντρικοποίησης και τη δομή των δικτύων.

1.7 Μη δομημένες αρχιτεκτονικές (Unstructured Architectures)

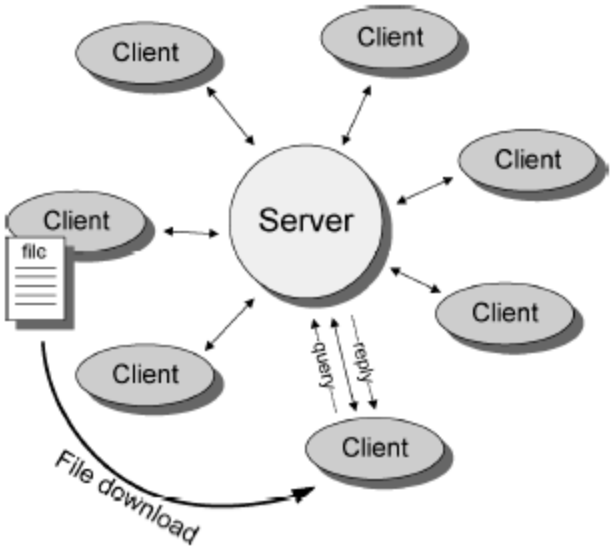
Υβριδικές Αποκεντρικοποιημένες (Hybrid Decentralized)

Στην αρχιτεκτονική ενός υβριδικού αποκεντρικοποιημένου peer-to-peer συστήματος κάθε υπολογιστής-πελάτης αποθηκεύει το περιεχόμενό του που είναι διαμοιράσιμο με τους υπόλοιπους του δικτύου. Όλοι οι clients συνδέονται με έναν κεντρικό υπολογιστή καταλόγου (central directory server) που διατηρεί:

- έναν πίνακα με πληροφορίες σύνδεσης των καταχωρημένων χρηστών (π.χ. διεύθυνση IP, το εύρος ζώνης της σύνδεσης)
- έναν πίνακα ο οποίος περιέχει μια λίστα με τα αρχεία που διατηρεί και διαμοιράζεται ο κάθε χρήστης, καθώς και τις περιγραφές των αρχείων σε μορφή μεταδεδομένων (π.χ. όνομα αρχείου, χρόνος δημιουργίας)

Κάθε φορά που ένας υπολογιστής επιθυμεί να συνδεθεί στο δίκτυο έρχεται σε επαφή με τον κεντρικό υπολογιστή και τον ενημερώνει για τα αρχεία που διατηρεί.

Όταν οι clients αναζητούν ένα αρχείο στέλνουν αιτήματα στον server. Ο κεντρικός υπολογιστής ψάχνει για τους αντίστοιχους δείκτες των αρχείων αυτών και επιστρέφει μια λίστα χρηστών, οι οποίοι περιέχουν το αρχείο που αναζητείται. Ο χρήστης ανοίγει έπειτα απ' ευθείας συνδέσεις με έναν ή περισσότερους από τους peers που περιέχουν το ζητούμενο αρχείο και το "κατεβάζει". Αυτή η διαδικασία απεικονίζεται στην Εικόνα 2.



Εικόνα 2: Μια αρχιτεκτονική ενός αποκεντρικοποιημένου υβριδικού peer-to-peer συστήματος. Ένας κεντρικός υπολογιστής καταλόγου (central directory server) διατηρεί έναν δείκτη των metadata για όλους τα αρχεία του δικτύου.

Το *πλεονέκτημα* των αποκεντρικοποιημένων υβριδικών συστημάτων είναι ότι είναι απλά στην υλοποίηση και εντοπίζουν τα αρχεία γρήγορα και αποδοτικά. Το κύριο *μειονέκτημά* τους είναι ότι είναι ευπαθή στον αυστηρό έλεγχο (censorship), τη νόμιμη δράση, την εποπτεία, την κακόβουλη επίθεση και την τεχνική αποτυχία, επειδή το κατανεμημένο περιεχόμενο, ακόμα και οι περιγραφές του, αλλά και η δυνατότητα πρόσβασης σ' αυτό ελέγχεται από μια επιχείρηση ή έναν χρήστη, οι οποίοι κατέχουν τον κεντρικό υπολογιστή. Επιπλέον, αυτά τα συστήματα θεωρούνται ότι δεν έχουν ικανότητα κλιμάκωσης (unscalable), διότι υπάρχουν περιορισμοί στο μέγεθος της βάσης δεδομένων των κεντρικών υπολογιστών και της χωρητικότητας τους να αποκριθούν στις ερωτήσεις. Οι μεγάλες μηχανές αναζήτησης ιστού αποτελούν, ωστόσο, την εξαίρεση στον κανόνα με αυτήν την έννοια. Τα παραδείγματα των hybrid decentralized content distribution συστημάτων περιλαμβάνουν τα συστήματα Napster και Publius, τα οποία στηρίζονται σε μια στατική λίστα κεντρικών υπολογιστών εύρους συστήματος. Η αρχιτεκτονική τους δεν παρέχει μια ομαλή, αποκεντρικοποιημένη υποστήριξη κατά την προσθήκη ενός νέου κεντρικού υπολογιστή ή την αφαίρεση των κεντρικών υπολογιστών, στα οποία έχουν γίνει κακόβουλες επιθέσεις.

Πρέπει να σημειωθεί ότι τα συστήματα που δεν ανήκουν στην υβριδική αποκεντρικοποιημένη κατηγορία μπορούν να χρησιμοποιήσουν κάποιον κεντρικό υπολογιστή σε περιορισμένη έκταση, π.χ. για το αρχικό bootstrapping συστήματος (π.χ. MojoNation [MojoNation, 2003]), ή για την άδεια σύνδεσης νέων χρηστών στο δίκτυο με

την παροχή πρόσβασης σε έναν κατάλογο υπαρχόντων χρηστών (π.χ. gnutellahosts.com για το δίκτυο gnutella).

Πλήρως αποκεντρικοποιημένα συστήματα

Σε αυτό το τμήμα, εξετάζουμε το δίκτυο Gnutella [G2003], το οποίο ανήκει στις purely decentralized peer-to-peer αρχιτεκτονικές, λόγω της ανοικτής αρχιτεκτονικής του, της διαπιστωμένης κλιμάκωσής του και της ικανότητας να προσαρμόζεται αυτόματα. Το FreeHaven [DFM2000] είναι ένα άλλο σύστημα που χρησιμοποιεί μηχανισμούς δρομολόγησης και αναζήτησης παρόμοιους με εκείνους του Gnutella.

Όπως τα περισσότερα peer-to-peer συστήματα, το Gnutella χτίζει ένα ιδεατό (virtual) overlay δίκτυο με τους δικούς του μηχανισμούς δρομολόγησης [RF2002], επιτρέποντας στους χρήστες του να μοιράζονται τα αρχεία με άλλους peers.

Δεν υπάρχει κανένας κεντρικός συντονισμός των ενεργειών στο δίκτυο και οι χρήστες συνδέονται ο ένας με τον άλλον άμεσα, μέσω ενός λογισμικού εφαρμογής που λειτουργεί και ως πελάτης και ως κεντρικός υπολογιστής. Αυτοί οι χρήστες αναφέρονται ως ***servents***.

Το Gnutella χρησιμοποιεί το IP ως τη βασική υπηρεσία δικτύου ενώ η επικοινωνία μεταξύ servents ορίζεται σε μια μορφή πρωτοκόλλου επιπέδου εφαρμογής υποστηρίζοντας τέσσερις τύπους μηνυμάτων [J2001]:

Ping: Ένα αίτημα για έναν host να δηλώσει την ύπαρξή του.

Pong: Απάντηση σε ένα μήνυμα Ping, το οποίο περιέχει την IP και τη θύρα του κόμβου που αποκρίνεται καθώς και τον αριθμό και το μέγεθος των διαμοιραζόμενων αρχείων.

Query: Ένα αίτημα αναζήτησης. Περιέχει το κείμενο αναζήτησης και τις ελάχιστες απαιτήσεις σε ταχύτητα του αποκρινόμενου host.

Query Hits: Απάντηση σε ένα Query μήνυμα. Περιέχει την IP, τη θύρα και την ταχύτητα απόκρισης του host καθώς και τον αριθμό των αρχείων που βρέθηκαν και το σύνολο των δεικτών τους.

Μετά την σύνδεση ενός node στο δίκτυο Gnutella, ο κόμβος στέλνει ένα *Ping* μήνυμα σε οποιοδήποτε κόμβο είναι συνδεμένος. Έπειτα, οι κόμβοι αυτοί στέλνουν ένα μήνυμα *Pong* που προσδιορίζουν μ' αυτό τον τρόπο τον εαυτό τους και επίσης διαδίδουν το μήνυμα *Ping* στους γειτονικούς nodes.

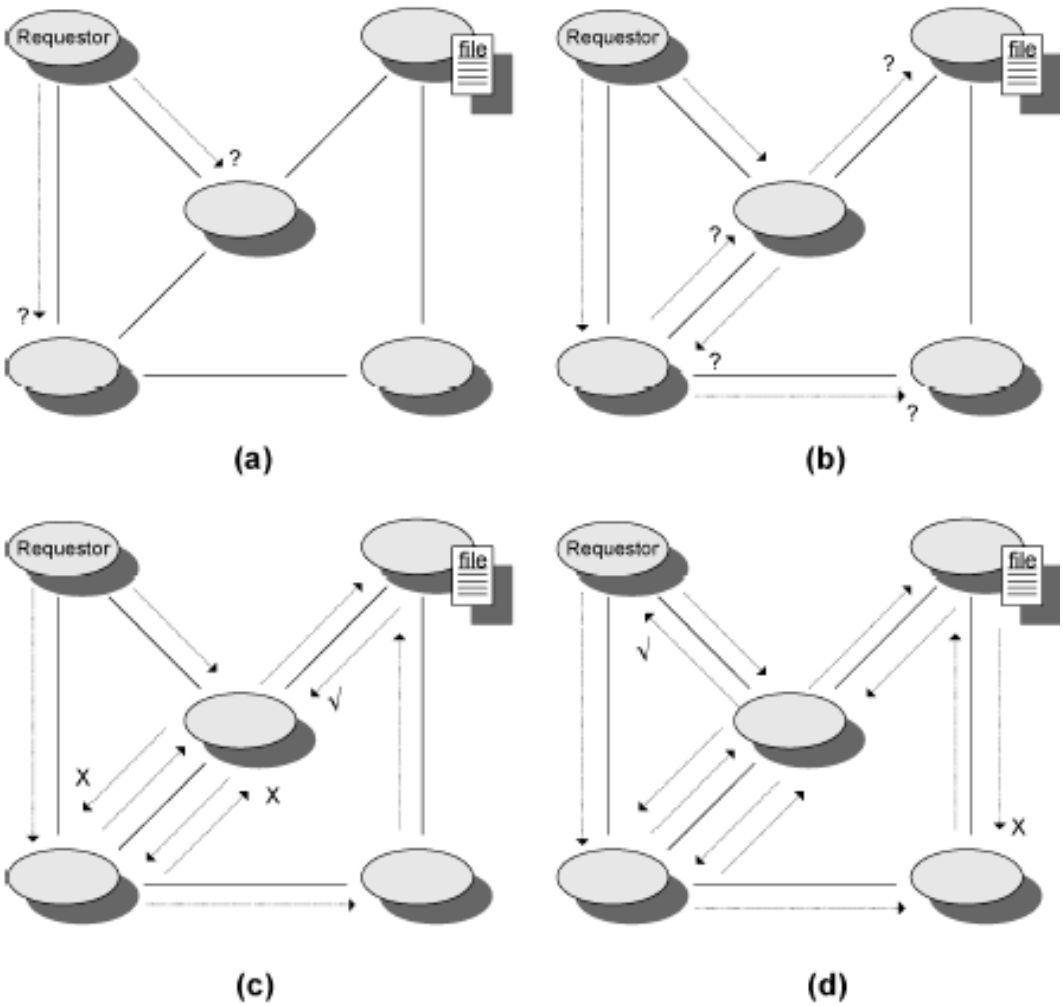
Προκειμένου να τοποθετηθεί ένα αρχείο σε ένα μη δομημένο σύστημα όπως το gnutella, η μόνη επιλογή είναι οι μη ντετερμινιστικές αναζητήσεις επειδή οι κόμβοι δεν έχουν κανέναν τρόπο να βρουν τα αρχεία που ψάχνουν.

Οι αρχικές αρχιτεκτονικές Gnutella χρησιμοποιούν μηχανισμούς καθολικής εκπομπής για να διανείμουν *Ping* και *Query* μηνύματα: κάθε Gnutella node διαβιβάζει τα

λαμβανόμενα μηνύματα σε όλους τους γειτονικούς του κόμβους. Η απάντηση στα λαμβανόμενα μηνύματα δρομολογείται στο σημείο από όπου έγινε το αρχικό αίτημα. Για να περιοριστεί η διάδοση μηνυμάτων μέσω του δικτύου, σε κάθε επικεφαλίδα μηνύματος περιέχεται ένα πεδίο που ονομάζεται χρόνος ζωής (TTL). Σε κάθε hop, η τιμή αυτού του πεδίου μειώνεται και όταν φθάσει στη τιμή μηδέν, το μήνυμα απορρίπτεται.

Ο παραπάνω μηχανισμός εφαρμόζεται με την ανάθεση μιας μοναδικής ταυτότητας (identifier) σε κάθε μήνυμα και εξοπλίζοντας κάθε host με έναν δυναμικό πίνακα δρομολόγησης που περιέχει τις ταυτότητες μηνυμάτων και τις διευθύνσεις των κόμβων. Επειδή τα μηνύματα απόκρισης περιέχουν το ίδιο ID (ταυτότητα) με τα αρχικά μηνύματα, ο host ελέγχει τον πίνακα δρομολόγησής του για να καθορίσει κατά μήκος ποιας σύνδεσης θα προωθήσει το μήνυμα απόκρισης. Για να αποφευχθούν οι βρόχοι (loops), οι κόμβοι χρησιμοποιούν τις μοναδικές ταυτότητες μηνυμάτων για να ανιχνεύσουν και να αποβάλλουν τα διπλά μηνύματα, για να βελτιώσουν την αποδοτικότητα και να διατηρήσουν το εύρος ζώνης των δικτύων [J2000].

Μόλις ένας κόμβος λάβει ένα μήνυμα Query Hit, δηλώνει ότι το αρχείο που αναζητούσε έχει βρεθεί σε έναν κόμβο και αρχίζει να δημιουργεί μια άμεση σύνδεση με το κόμβο αυτό. Η Εικόνα 3 εξηγεί ένα παράδειγμα μηχανισμού αναζήτησης Gnutella. Οι γραμμές μεταξύ των κόμβων απεικονίζουν την σύνδεση του Gnutella δικτύου. Η αναζήτηση ξεκινά από τον κόμβο που έκανε την αίτηση. Τα μηνύματα αίτησης αποστέλλονται σε όλους τους γειτονικούς κόμβους, και διαδίδονται από κόμβο σε κόμβο όπως φαίνεται στα τέσσερα διαδοχικά βήματα από το (a) έως το (d).



Εικόνα 3: Ένα παράδειγμα του Gnutella μηχανισμού αναζήτησης.

Ζητήματα κλιμάκωσης στα purely decentralized συστήματα προέκυψαν από το γεγονός ότι η χρήση του TTL διαιρεί αποτελεσματικά το δίκτυο σε "sub networks" (υποδίκτυα), επιβάλλοντας σε κάθε χρήστη έναν εικονικό "ορίζοντα" πέρα από τον οποίο τα μηνύματά τους δεν θα μπορούν να είναι προσιτά [J2000]. Από την άλλη μεριά αφαίρεση του TTL θα οδηγούσε το δίκτυο σε συμφόρηση από μηνύματα.

Μερικώς Κεντρικοποιημένα (Partially Centralized)

Τα partially centralized συστήματα είναι παρόμοια με τα purely decentralized αλλά χρησιμοποιούν την έννοια των ***supernodes***. Ως supernodes ορίζονται οι κόμβοι που έχουν στόχο την δεικτοδότηση (indexing) και την ενδιάμεση αποθήκευση (caching) των αρχείων που υπάρχουν στο δίκτυο.

Οι peers επιλέγονται αυτόματα για να γίνουν supernodes εάν έχουν ικανοποιητικό εύρος ζώνης και επεξεργαστική ισχύ (αν και μια παράμετρος διαμόρφωσης μπορεί να επιτρέψει στους χρήστες να θέσουν εκτός λειτουργίας αυτό το χαρακτηριστικό γνώρισμα) [F2003].

Οι supernodes τοποθετούν σ' έναν κατάλογο τα διαμοιραζόμενα αρχεία που συνδέονται με τους peers και η proxy αναζήτηση γίνεται μέσω αυτών των peers. Όλες οι ερωτήσεις, επομένως, κατευθύνονται αρχικά στους supernodes.

Δύο σημαντικά *πλεονεκτήματα* των μερικώς κεντρικοποιημένων συστημάτων είναι τα εξής:

- Ο χρόνος ανακάλυψης μειώνεται σε σύγκριση με τα purely decentralized συστήματα, ενώ ακόμα δεν υπάρχει κανένα σημείο αποτυχίας. Εάν ένας ή περισσότεροι supernodes καταστραφούν τότε οι κόμβοι που συνδέονται με αυτούς μπορούν να δημιουργήσουν νέες συνδέσεις με άλλους supernodes και το δίκτυο θα συνεχίσει να λειτουργεί κανονικά.
- Η έμφυτη ετερογένεια ενός peer-to-peer δικτύου είναι ένα πλεονέκτημα προς εκμετάλλευση. Σ' ένα purely decentralized δίκτυο, όλοι οι κόμβοι θα είναι εξίσου φορτωμένοι, ανεξάρτητα από τη δύναμη της ΚΜΕ τους, το εύρος ζώνης ή τις ικανότητες αποθήκευσης. Στα partially centralized συστήματα, εντούτοις, οι supernodes θα αναλάβουν μια μεγάλη μερίδα ολόκληρου του φορτίου του δικτύου, ενώ το μεγαλύτερο μέρος των κόμβων, οι οποίοι ονομάζονται "normal", θα είναι πολύ ελαφριά φορτωμένοι [LRS2002, ZMK2002].

Το Kazaa είναι μια χαρακτηριστική περίπτωση των partially centralized systems (δεδομένου ότι είναι ένα ιδιόκτητο σύστημα, δεν υπάρχει καμία λεπτομερής τεκμηρίωση στη δομή και λειτουργία του). Το Edutella [NWDSNN2003] είναι μια άλλη partially centralized αρχιτεκτονική.

Η έννοια των supernodes παρουσιάζεται, επίσης, σε μια πιο πρόσφατη έκδοση του πρωτοκόλλου Gnutella. Ένας μηχανισμός για την επιλογή supernodes οργανώνει το δίκτυο Gnutella σε μια διασύνδεση από *superpeers* και κόμβους πελάτες.

Όταν ένας κόμβος με αρκετή υπολογιστική ισχύ εισέρχεται στο δίκτυο, γίνεται αμέσως ένας *superpeer* και εγκαθιστά τις συνδέσεις με άλλα superpeers, δημιουργώντας ένα μη δομημένο επίπεδο δίκτυο από superpeers. Εάν εγκαταστήσει έναν ελάχιστο αριθμό από συνδέσεις με κόμβους πελάτες μέσα σε ένα καθορισμένο χρόνο, παραμένει ένας *superpeer*. Διαφορετικά, μετατρέπεται σε κανονικό κόμβο πελάτη.

1.8 Δομημένες Αρχιτεκτονικές

Τα διάφορα δομημένα content distribution συστήματα και οι υποδομές τους χρησιμοποιούν διαφορετικούς μηχανισμούς για τη δρομολόγηση των μηνυμάτων και τη

θέση των δεδομένων. Τέσσερις από τους πιο ενδιαφέροντες και αντιπροσωπευτικούς μηχανισμούς με τα αντίστοιχα συστήματά τους εξετάζονται παρακάτω.

- Το *Freenet* είναι ένα *χαλαρά δομημένο* σύστημα που χρησιμοποιεί τη μέθοδο της ομοιότητας με identifiers αρχείων και των κόμβων για να υπολογίσει την τοποθεσία ενός αρχείου. Επίσης, εφαρμόζει την προσέγγιση της προώθησης με αλυσίδα (chain mode propagation) για τις ερωτήσεις που προωθούνται από κόμβο σε κόμβο.
- Το *Chord* είναι ένα σύστημα του οποίου οι κόμβοι διατηρούν έναν κατανεμημένο πίνακα δρομολόγησης, υπό μορφή ενός κύκλου από identifier στον οποίο όλοι οι κόμβοι χαρτογραφούνται, και ένα πίνακα δεικτών (*finger table*).
- Το *CAN* είναι ένα σύστημα το οποίο χρησιμοποιεί ένα Καρτεσιανό χώρο n-διαστάσεων για να εφαρμόσει τον κατανεμημένο πίνακα θέσης και δρομολόγησης, μέσω του οποίου κάθε κόμβος είναι αρμόδιος για μια ζώνη *(zone)* στο ισοδύναμο διάστημα.
- Το *Tapestry* είναι βασισμένο στη δομή δεδομένων *plaxton mesh*, η οποία διατηρεί τους δείκτες των κόμβων στο δίκτυο, των οποίων τα IDs ταιριάζουν με τα στοιχεία μιας δομής δέντρου που περιέχει προθέματα ID μέχρι μια θέση ψηφίων.

Freenet–Ένα χαλαρά δομημένο σύστημα

Το χαρακτηριστικό των χαλαρά δομημένων συστημάτων είναι ότι οι κόμβοι του peer-to-peer δικτύου μπορούν να κάνουν εκτιμήσεις (όχι με βεβαιότητα) για τον κόμβο που μπορεί να αποθηκεύσει το περιεχόμενο. Αυτό τους δίνει τη δυνατότητα να αποφύγουν την καθολική εκπομπή μηνυμάτων αίτησης σε όλους τους γείτονές τους ή σ' ένα τυχαίο υποσύνολό τους. Αντί αυτού, χρησιμοποιούν την προσέγγιση chain mode propagation, όπου κάθε κόμβος λαμβάνει τοπικά μια απόφαση για το ποιος κόμβος στέλνει το μήνυμα αίτησης στον επόμενο.

Το Freenet [Clarke et αl., 2000] είναι ένα χαρακτηριστικό purely decentralized χαλαρά δομημένο κατανεμημένο σύστημα. Λειτουργεί ως self-organizing peer-to-peer δίκτυο, συγκεντρώνοντας τον αχρησιμοποίητο χώρο δίσκου στους peers για να δημιουργήσει ένα συνεργάσιμο εικονικό σύστημα αρχείων. Τα σημαντικά χαρακτηριστικά γνωρίσματα του Freenet περιλαμβάνουν την ασφάλεια, την ανωνυμία δημοσίευσης (anonymity publisher), την δυνατότητα άρνησης (deniability), τη δυνατότητα δημιουργίας αντιγράφων για τα δεδομένα (data replication) με σκοπό την αυξημένη διαθεσιμότητα και απόδοση.

Τα αρχεία στο Freenet προσδιορίζονται από μοναδικά δυαδικά κλειδιά. Τρεις τύποι κλειδιών υπάρχουν. Ο απλούστερος είναι βασισμένος στην εφαρμογή μιας hash συνάρτησης σε ένα περιγραφικό κείμενο που συνοδεύει κάθε αρχείο καθώς αυτό αποθηκεύεται στο δίκτυο από τον αρχικό ιδιοκτήτη του.

Κάθε κόμβος του Freenet διατηρεί τα δεδομένα του, τα οποία τα καθιστά διαθέσιμα στο δίκτυο για ανάγνωση και εγγραφή, καθώς επίσης και έναν δυναμικό πίνακα δρομολόγησης που περιέχει τις διευθύνσεις και τα αρχεία άλλων κόμβων. Για την αναζήτηση ενός αρχείου, ο χρήστης στέλνει ένα μήνυμα αίτησης που καθορίζει την τιμή του *κλειδιού (key)* και το χρόνο *timeout* (hops to live).

Το Freenet χρησιμοποιεί τους ακόλουθους τύπους μηνυμάτων, που όλοι περιλαμβάνουν τον identifier του κόμβου (για την ανίχνευση επαναλήψεων), την τιμή hopes to live (παρόμοια με το Gnutella TTL) και τα identifiers των κόμβων της πηγής και του προορισμού:

Data insert (Εισαγωγή δεδομένων)

Ένα data insert μήνυμα στέλνεται όταν ένας κόμβος εισάγει νέα στοιχεία στο δίκτυο. Στο μήνυμα αυτό περιλαμβάνεται το κλειδί και τα αρχεία.

Data request (Αίτηση εύρεσης δεδομένων)

Ένα data request μήνυμα στέλνεται όταν γίνεται αίτηση για ένα ορισμένο αρχείο. Στο μήνυμα συμπεριλαμβάνεται και το κλειδί του ζητούμενου αρχείου.

Data reply (Απάντηση ερώτησης εύρεσης δεδομένων)

Μια απάντηση ξεκινά όταν το ζητούμενο αρχείο εντοπισθεί. Το αρχείο αυτό συμπεριλαμβάνεται στο μήνυμα απάντησης.

Data failed (Αποτυχία εύρεσης δεδομένων)

Ένα data failed μήνυμα στέλνεται στην περίπτωση αποτυχίας εντοπισμού ενός αρχείου. Στο μήνυμα συμπεριλαμβάνονται η θέση (κόμβος) και ο λόγος της αποτυχίας.

Οι νέοι κόμβοι συνδέονται στο Freenet δίκτυο έχοντας πρώτα ανακαλύψει τη διεύθυνση ενός ή περισσότερων υπαρχόντων κόμβων, και έπειτα αρχίζουν να στέλνουν τα *Data Insert* μηνύματα. Για την εισαγωγή ενός νέου αρχείου στο δίκτυο, ο κόμβος υπολογίζει αρχικά ένα δυαδικό κλειδί για το αρχείο και στέλνει έπειτα ένα data insert μήνυμα. Οποιοσδήποτε κόμβος λάβει το data insert μήνυμα, ελέγχει πρώτα εάν το κλειδί υπάρχει ήδη. Εάν το κλειδί δεν βρεθεί, ο κόμβος ανατρέχει στο πιο "κοντινό" κλειδί (από άποψη λεξικογραφικής απόστασης) στον πίνακα δρομολόγησής του και προωθεί το data insert μήνυμα στον αντίστοιχο κόμβο. Σύμφωνα μ' αυτόν τον μηχανισμό, τα νέα αρχεία τοποθετούνται στους κόμβους που έχουν αρχεία με παρόμοια κλειδιά.

Αυτό συνεχίζεται εφ' όσον δεν παραβιάζεται το όριο hops to live. Κατ' αυτό τον τρόπο, περισσότεροι από ένας κόμβοι θα αποθηκεύσουν το νέο αρχείο. Συγχρόνως, όλοι οι συμμετέχοντες κόμβοι θα ενημερώσουν τους πίνακες δρομολόγησής τους με τις νέες πληροφορίες. Αυτός είναι ο μηχανισμός μέσω του οποίου οι νέοι κόμβοι αναγγέλλουν την παρουσία τους στο υπόλοιπο δίκτυο. Εάν το hopes to live όριο φτάνεται χωρίς

σύγκρουση, ένα “all clear” αποτέλεσμα θα διαδοθεί πίσω στον αρχικό κόμβο (inserter), ενημερώνοντας τον ότι το insert μήνυμα ήταν επιτυχές.

Εάν το κλειδί βρεθεί, ο κόμβος επιστρέφει το προϋπάρχων αρχείο σαν να έχει γίνει αίτηση για αυτό. Κατ' αυτό τον τρόπο, οι κακόβουλες προσπάθειες να αντικατασταθούν τα υπάρχοντα αρχεία με την παρεμβολή junk θα οδηγήσουν σε μεγαλύτερη διάδοση υπαρχόντων αρχείων.

Εάν ένας κόμβος λάβει μια αίτηση για ένα αρχείο που είναι αποθηκευμένο τοπικά, η αναζήτηση σταματά και το αρχείο προωθείται στον κόμβο που έκανε την αίτηση (requestor).

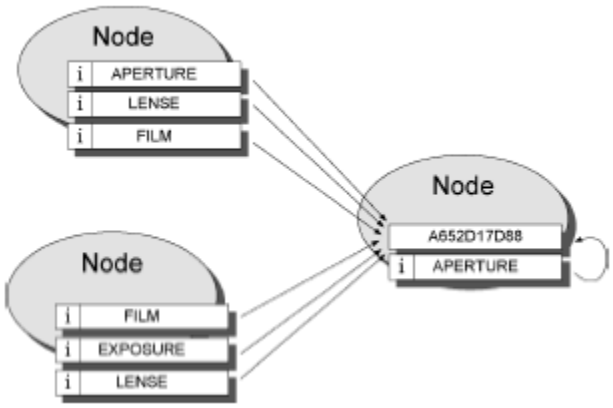
Εάν ο κόμβος δεν διαθέτει το αρχείο που ο requestor αναζητά, προωθεί την αίτηση στο γείτονά του που είναι πιθανό να έχει το αρχείο, αναζητώντας το κλειδί του αρχείου στον τοπικό πίνακα δρομολόγησής του. Τα μηνύματα, επομένως, διαμορφώνουν μια αλυσίδα, καθώς διαδίδονται από κόμβο σε κόμβο (node to node). Για να αποφευχθούν οι τεράστιες αλυσίδες, τα μηνύματα διαγράφονται μετά το πέρασμα ορισμένων nodes, σύμφωνα με τη hops to live τιμή που έχουν. Οι κόμβοι αποθηκεύουν, επίσης, την ταυτότητα (ID) και άλλες πληροφορίες των αιτήσεων, προκειμένου να καθοδηγούν τα “Data reply” και τα “Data failed” μηνύματα.

Εάν ένας κόμβος λαμβάνει ένα backtracking “data failed” μήνυμα από έναν downstream κόμβο, επιλέγει τον επόμενο καλύτερο κόμβο από τη σωρό δρομολόγησής του και προωθεί την αίτηση σε αυτόν. Εάν όλοι οι nodes στον πίνακα δρομολόγησης έχουν εξερευνηθεί κατ' αυτό τον τρόπο και έχουν αποτύχει, στέλνει πίσω ένα “*data failed*” μήνυμα στον κόμβο από τον οποίο έλαβε αρχικά το μήνυμα αίτησης.

Εάν το ζητούμενο αρχείο βρεθεί, τότε ο αρχικός κόμβος λαμβάνει μια απάντηση μέσω των κόμβων που διαβίβασαν το αίτημα. Αυτό το *data reply* μήνυμα θα περιέχει το αρχείο, το οποίο αποθηκεύεται σε όλους τους ενδιάμεσους κόμβους για μελλοντικές αιτήσεις. Στη συνέχεια, μια αίτηση με το ίδιο κλειδί θα εξυπηρετηθεί αμέσως από το αποθηκευμένο αρχείο. Μια αίτηση για ένα παρόμοιο κλειδί θα διαβιβαστεί στον κόμβο που παρείχε προηγουμένως τα δεδομένα.

Για να λυθεί το πρόβλημα της απόκτησης του κλειδιού, το οποίο αντιστοιχίζεται σε ένα συγκεκριμένο αρχείο, το Freenet προτείνει τη χρήση μιας ειδικής κατηγορίας αρχείων αποκαλούμενων “έμμεσα αρχεία” (indirect files). Όταν ένα πραγματικό αρχείο εισάγεται, ο δημιουργός του αρχείου εισάγει επίσης και διάφορα έμμεσα αρχεία που ονομάζονται σύμφωνα με τις λέξεις-κλειδιά αναζήτησης και περιέχουν τους δείκτες του πραγματικού αρχείου. Αυτά τα έμμεσα αρχεία διαφέρουν από τα κανονικά στο ότι πολλαπλά αρχεία με το ίδιο κλειδί (δηλ. Λέξη-κλειδί αναζήτησης) επιτρέπονται να υπάρχουν, και οι αιτήσεις για τέτοια κλειδιά συνεχίζουν έως ότου συσσωρευτεί ένας αριθμός αποτελεσμάτων, αντί να σταματά η αναζήτηση στην εύρεση του πρώτου αρχείου. Το πρόβλημα της διαχείρισης τέτοιου μεγάλου όγκου έμμεσων αρχείων παραμένει ανοικτό.

Η Εικόνα 4 δείχνει τη χρήση των έμμεσων αρχείων. Το διάγραμμα απεικονίζει ένα αρχείο με κλειδί "A652D17D88". Η εισαγωγή γίνεται τυχαία, και τότε ένα σύνολο από έμμεσα αρχεία (σημειωμένα με ένα **i**) που ονομάζονται σύμφωνα με τις λέξεις-κλειδιά αναζήτησης. Τα έμμεσα αρχεία διανέμονται μεταξύ των κόμβων του δικτύου και περιέχουν έναν δείκτη προς το πραγματικό αρχείο.



Εικόνα 4: Η χρήση των έμμεσων αρχείων στο Freenet.

Οι ακόλουθες ιδιότητες του Freenet είναι ένα αποτέλεσμα των αλγορίθμων δρομολόγησης και τοποθέτησης:

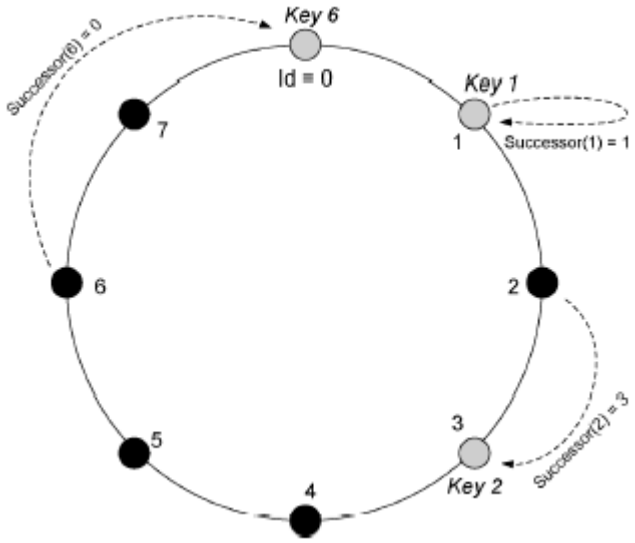
- Δεδομένου ότι οι κόμβοι παίρνουν τις ερωτήσεις από άλλους κόμβους με παρόμοια κλειδιά, τείνουν να ειδικευτούν στην αναζήτηση για παρόμοια κλειδιά κατά τη διάρκεια του χρόνου.
- Οι κόμβοι αποθηκεύουν τα παρόμοια κλειδιά, λόγω της ενδιάμεσης αποθήκευσης (cashing) των αρχείων.
- Η ομοιότητα των κλειδιών δεν απεικονίζει την ομοιότητα των αρχείων.
- Η δρομολόγηση δεν απεικονίζει την τοπολογία δικτύων.

Chord

Το Chord [Stoica et al., 2001] είναι μια peer-to-peer υποδομή δρομολόγησης και τοποθέτησης, που εκτελεί μια αντιστοίχηση των identifiers των αρχείων στους identifiers των κόμβων. Η τοποθέτηση των δεδομένων μπορεί να εφαρμοστεί στην κορυφή του Chord ταυτοποιώντας δεδομένα (αρχεία) με κλειδιά και την αποθήκευση των *(κλειδί, data item)* ζευγαριών στον κόμβο στον οποίο τα κλειδιά αντιστοιχήθηκαν.

Στο Chord, οι κόμβοι προσδιορίζονται επίσης από κλειδιά. Τα κλειδιά δίνονται και στα αρχεία και στους κόμβους με τη βοήθεια μιας ντετερμινιστικής συνάρτησης, μια

παραλλαγή της consistent hashing [KLLLLP1997]. Όλοι οι identifiers των κόμβων τοποθετούνται σε έναν "κύκλο ταυτοτήτων" (identifier circle) modulo 2^m. Στην Εικόνα 5 παρουσιάζεται ένας τέτοιος κύκλος με m=3. Το κλειδί k ορίζεται στον πρώτο κόμβο του οποίου ο identifier είναι ίσος ή ακολουθεί το k στο χώρο που ορίζουν οι identifiers. Αυτός ο κόμβος καλείται *successor* του κλειδιού k. Η χρήση consistent hashing τείνει να εξισορροπήσει το φορτίο, δεδομένου ότι κάθε κόμβος λαμβάνει κατά προσέγγιση τον ίδιο αριθμό κλειδιών.



Εικόνα 5: Ένας Chord "κύκλος ταυτοτήτων" που αποτελείται από τρεις κόμβους 0,1 και 3. Το κλειδί 1 τοποθετείται στον κόμβο 1, το κλειδί 2 στον 3 και το κλειδί 6 στον 0.

Οι μόνες πληροφορίες δρομολόγησης που απαιτούνται είναι να γνωρίζει ο κάθε κόμβος τον successor του στον κύκλο. Οι ερωτήσεις για ένα δεδομένο κλειδί περνούν πάνω από τον κύκλο μέσω των successor δεικτών έως ότου βρεθεί ο κόμβος που περιέχει το κλειδί.

Όταν ένας κόμβος *n* ενωθεί στο δίκτυο, ορισμένα κλειδιά που έχουν ανατεθεί στον successor του n, θα ανατεθούν τώρα στον *n*. Όταν ο κόμβος *n* αποσυνδεθεί από το δίκτυο, όλα τα κλειδιά που ανατέθηκαν σε αυτόν θα ανατεθούν εκ νέου στον successor του. Αυτές είναι οι μόνες αλλαγές στις αναθέσεις των κλειδιών που πρέπει να πραγματοποιηθούν προκειμένου να διατηρηθεί η εξισορρόπηση του φορτίου.

Όπως αναφέρθηκε, μόνο ένα στοιχείο ανά κόμβο θα πρέπει να είναι σωστό για το Chord για να εγγυάται σωστή δρομολόγηση των ερωτήσεων. Η απόδοση μειώνεται όταν οι πληροφορίες δρομολόγησης γίνονται μη συνεπείς εξ αιτίας της σύνδεσης και της αποσύνδεσης των κόμβων από το σύστημα, και η διαθεσιμότητα παραμένει υψηλή μόνο εφ' όσον οι κόμβοι αποσυνδέονται ανεξάρτητα. Δεδομένου ότι η overlay τοπολογία δεν

είναι βασισμένη στην φυσική τοπολογία δικτύων IP, μια αποτυχία στο δίκτυο IP μπορεί να δημιουργήσει πολλαπλές αποτυχίες συνδέσεων στο overlay δίκτυο [SGG2002].

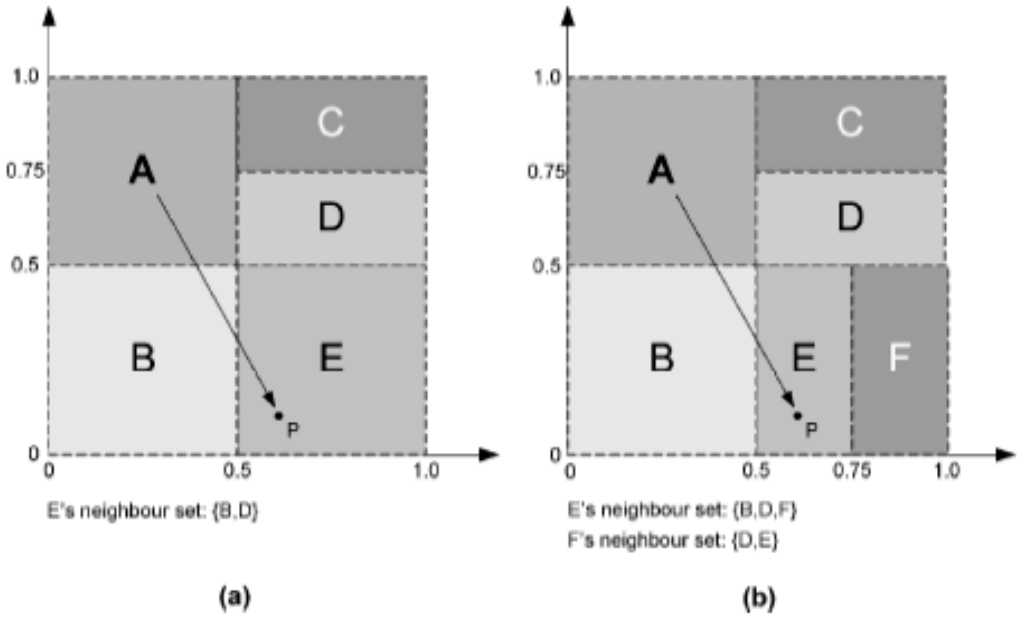
Για να αυξηθεί η αποδοτικότητα του μηχανισμού τοποθέτησης, που περιγράφηκε προηγουμένως, το Chord διατηρεί τις πρόσθετες πληροφορίες δρομολόγησης, υπό μορφή "finger table". Σε αυτόν τον πίνακα, κάθε είσοδος *i* οδηγεί στο successor του κόμβου $n+2^i$. Για να γίνει μια διαδικασία αναζήτησης (lookup) του κλειδιού *k* για έναν κόμβο *n*, το finger table προσδιορίζει τον υψηλότερο κόμβο *n'*, του οποίου το ID είναι μεταξύ του *n* και του *k*. Εάν ένας τέτοιος κόμβος υπάρχει, η αναζήτηση επαναλαμβάνεται αρχικά από το *n'*. Διαφορετικά, ο successor του *n* επιστρέφεται. Χρησιμοποιώντας το finger table, τόσο το πλήθος των πληροφοριών δρομολόγησης που υπάρχει σε κάθε κόμβο, και ο απαιτούμενος χρόνος για την επίλυση των lookups είναι *O(logN)* για ένα N - σύστημα κόμβων σε σταθερή κατάσταση.

CAN

Το CAN ("Content Addressable Network") [RFHK2001] είναι ουσιαστικά ένας distributed, Internet-scale hash πίνακας που αντιστοιχίζει ονόματα αρχείων σε θέσεις στο δίκτυο, και υποστηρίζει τις λειτουργίες της εισαγωγής, της αναζήτησης (lookup) και της διαγραφής ζευγαριών *(κλειδί, τιμή)* στον πίνακα.

Κάθε κόμβος του δικτύου CAN αποθηκεύει ένα μέρος, το οποίο καλείται "ζώνη" (zone), του hash πίνακα καθώς επίσης και τις πληροφορίες για έναν μικρό αριθμό γειτονικών ζωνών στον πίνακα. Οι αιτήσεις για εισαγωγή, αναζήτηση ή διαγραφή ενός κλειδιού, δρομολογούνται μέσω των ενδιάμεσων ζωνών στον κόμβο που διατηρεί τη ζώνη που περιέχει το κλειδί.

Το CAN χρησιμοποιεί ένα εικονικό d-διαστάσεων Καρτεσιανό χώρο (Εικόνα 6) για να αποθηκεύει τα ζευγάρια *(κλειδί K, value V)*. Κάθε κόμβος αντιστοιχεί σε ένα τμήμα αυτού του διαστήματος που αποτελεί τη ζώνη του hash πίνακα. Κάθε κλειδί *k* χαρτογραφείται επάνω σε ένα σημείο *P* στο χώρο. Το (*K, V*) ζευγάρι αποθηκεύεται έπειτα στον κόμβο που είναι αρμόδιος για τη ζώνη μέσα στην οποία το σημείο *P* βρίσκεται. Παραδείγματος χάριν, στην περίπτωση που φαίνεται στην Εικόνα 6(a), ένα κλειδί που αντιστοιχεί στις συντεταγμένες *(0.1,0.2)* θα αποθηκευτεί στον κόμβο που είναι αρμόδιος για τη ζώνη *B*.



Εικόνα 6: (a) Ένα παράδειγμα με 2-d [0,1] x [0,1] χώρο χωρισμένο μεταξύ 5 CAN κόμβων (b) Ένα παράδειγμα 2-d χώρου μετά από την σύνδεση του κόμβου F.

Για να ανακτηθεί το δεδομένο που αντιστοιχεί στο κλειδί K, κάθε κόμβος μπορεί να χρησιμοποιήσει την ίδια ντετερμινιστική λειτουργία για να αντιστοιχήσει το *K* στο *P* και να ανακτήσει έπειτα την αντίστοιχη τιμή *V* από τον κόμβο που καλύπτει το *P*. Στην περίπτωση που το *P* βρεθεί στη ζώνη του requestor, η αίτηση πρέπει να καθοδηγηθεί από κόμβο σε κόμβο έως ότου φθάσει στον κόμβο που καλύπτει το *P*.

Οι CAN κόμβοι διατηρούν έναν πίνακα δρομολόγησης που περιέχει τις διευθύνσεις IP των nodes που έχουν τις γειτονικές ζώνες, για να επιτρέψει τη δρομολόγηση μεταξύ των αυθαίρετων σημείων στο χώρο. Η δρομολόγηση στο CAN επιτυγχάνεται ακολουθώντας την πορεία μέσω του Καρτεσιανού διαστήματος από την πηγή στον προορισμό. Παραδείγματος χάριν, στην Εικόνα 6(a), μια αίτηση από τον κόμβο *A* για ένα κλειδί που αντιστοιχίζεται στο σημείο *P* θα δρομολογηθεί μέσω των κόμβων A, B, E σύμφωνα με την ευθεία γραμμή που δείχνει το βέλος.

Ένας νέος κόμβος που συνδέεται στο CAN σύστημα ακολουθεί την εξής διαδικασία:

1. Ο νέος κόμβος προσδιορίζει έναν κόμβο που είναι ήδη μέσα στο CAN δίκτυο, χρησιμοποιώντας έναν bootstrap μηχανισμό, όπως περιγράφεται στο [Francis, 2000].
2. Χρησιμοποιώντας το μηχανισμό δρομολόγησης του CAN, ο κόμβος επιλέγει τυχαία ένα σημείο *P* στο διάστημα και στέλνει μια JOIN αίτηση στον κόμβο που καλύπτει το *P*. Η ζώνη έπειτα χωρίζεται και η μισή από αυτήν δίνεται στο νέο κόμβο.

3. Ο νέος κόμβος δημιουργεί τον πίνακα δρομολόγησής του με τις διευθύνσεις IP των νέων γειτόνων του και οι γείτονες της διασπασμένης ζώνης ενημερώνουν τους πίνακες δρομολόγησής τους για να συμπεριλάβουν το νέο κόμβο.

Όταν οι κόμβοι αποσυνδέονται από το CAN, οι ζώνες που τους έχουν ανατεθεί και οι hash πίνακες παραδίδονται σε έναν από τους γείτονές τους. Ένας κόμβος στέλνει περιοδικά μηνύματα αναπροσαρμογών σε κάθε γείτονά του αναφέροντας τις συντεταγμένες ζώνης του, τον κατάλογο και τις συντεταγμένες ζώνης των γειτόνων του. Εάν υπάρχει απουσία ενός τέτοιου μηνύματος αναπροσαρμογών, οι γειτονικοί κόμβοι αντιλαμβάνονται ότι έχει υπάρξει σφάλμα στον κόμβο αυτό και αρχίζουν έναν μηχανισμό ανάνηψης. Εάν πολλοί από τους γείτονες ενός κόμβου που έχει αποτύχει αποτυγχάνουν, ένας μηχανισμός αναζήτησης εκτεταμένου δακτυλίου (expanding ring) αρχίζει από έναν γειτονικό κόμβο, για να προσδιορίσει λειτουργικούς κόμβους έξω από την περιοχή αποτυχίας.

Tapestry

Το Tapestry [ZKJ2001] υποστηρίζει την τοποθέτηση των αντικειμένων και τη δρομολόγηση των μηνυμάτων προς αυτά (ή προς το κοντινότερο αντίγραφο εάν υπάρχουν περισσότερα από ένα) με κατανεμημένο, αυτό-διαχειριζόμενο και ανεκτικό σε σφάλματα τρόπο. Μ’ αυτόν τον τρόπο προσφέρεται σταθερότητα εύρους συστήματος καθώς επίσης και γρήγορη προσαρμογή των τοπολογιών επικοινωνίας σε διάφορες συνθήκες.

Η τοπολογία του δικτύου χαρακτηρίζεται ως προσαρμοστική καθώς οι κόμβοι συνδέονται ή αποσυνδέονται και οι συνδέσεις των δικτύων αποτυγχάνουν ή ανακτώνται. Οι πληροφορίες δρομολόγησης και τοποθέτησης κατανέμονται μεταξύ των κόμβων του δικτύου. Η συνέπεια της τοπολογίας ελέγχεται κατά την εκτέλεση, και εάν χαθεί λόγω αποτυχιών ή καταστραφεί, είναι εύκολο να ανασχηματιστεί ή να ανανεωθεί.

Το Tapestry είναι βασισμένο στους μηχανισμούς τοποθέτησης και δρομολόγησης που χρησιμοποιούνται στο Plaxton (RR1997), όπου αναφέρεται ως *Plaxton mesh*. Το Plaxton mesh αποτελεί μια κατανεμημένη δομή δεδομένων που επιτρέπει στους κόμβους να τοποθετήσουν τα αντικείμενα και τα μηνύματα δρομολόγησής τους μέσω ενός αυθαίρετα-ταξινομημένου overlay δικτύου, χρησιμοποιώντας χάρτες δρομολόγησης μικρού και σταθερού μεγέθους. Στο πρότυπο Plaxton mesh οι κόμβοι μπορούν να παίξουν το ρόλο των κεντρικών υπολογιστών (όπου τα αντικείμενα αποθηκεύονται), δρομολογητών (από όπου προωθούνται τα μηνύματα) και πελατών (από όπου προέρχονται οι αιτήσεις).

Κάθε κόμβος διατηρεί έναν *χάρτη γειτόνων (neighbor map)*, όπως φαίνεται στο παράδειγμα (Πίνακας 2). Ο neighbor map έχει πολλαπλά επίπεδα (levels). Κάθε επίπεδο *l* περιέχει τους δείκτες των κόμβων, των οποίων οι ταυτότητες πρέπει να αντιστοιχούν στα ψηφία *l* (τα x αντιπροσωπεύουν οποιαδήποτε ψηφία). Κάθε εγγραφή (entry) στο χάρτη

γειτόνων αντιστοιχεί σε έναν δείκτη του πιο κοντινού κόμβου στο δίκτυο του οποίου η ταυτότητα ταιριάζει με τον αριθμό στο χάρτη γειτόνων, μέχρι μια θέση ψηφίων.

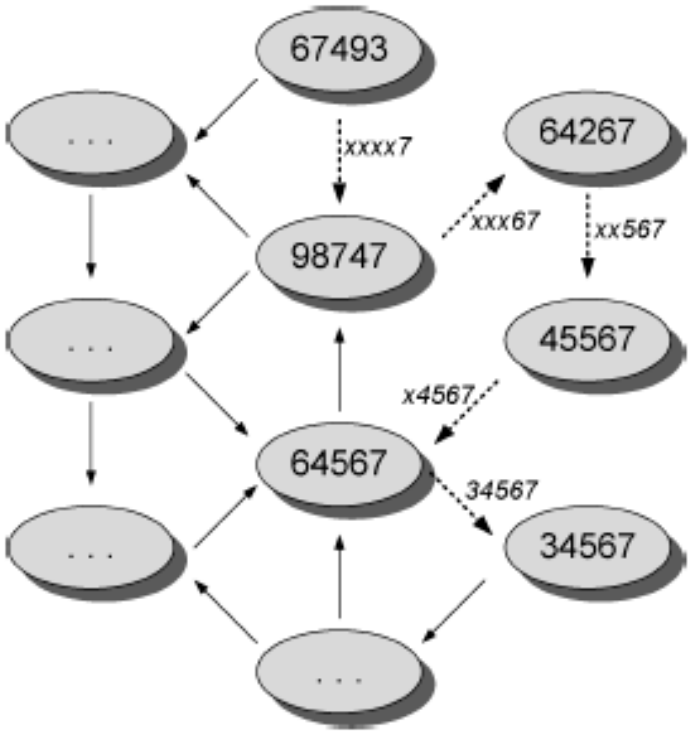
Παραδείγματος χάριν, η 5η εγγραφή για το 3ο επίπεδο για τον κόμβο 67493 του Πίνακας 2, οδηγεί σε κόμβο κοντά στο 67493, σε απόσταση δικτύων, του οποίου η ταυτότητα τελειώνει σε 593.

Πίνακας 2. Ο χάρτης που υπάρχει στον κόμβο με ID 67493

	Level 5	Level 4	Level 3	Level 2	Level 1
Entry 0	07493	x0493	xx093	xxx03	xxxx0
Entry 1	17493	x1493	xx193	xxx13	xxxx1
Entry 2	27493	x2493	xx293	xxx23	xxxx2
Entry 3	37493	x3493	xx393	xxx33	xxxx3
Entry 4	47493	x4493	xx493	xxx43	xxxx4
Entry 5	57493	x5493	xx593	xxx53	xxxx5
Entry 6	**67493**	x6493	xx693	xxx63	xxxx6
Entry 7	77493	x7493	xx793	xxx73	xxxx7
Entry 8	87493	x8493	xx893	xxx83	xxxx8
Entry 9	97493	x9493	xx993	xxx93	xxxx9

Τα μηνύματα, επομένως, δρομολογούνται στο προορισμό τους ψηφίο προς ψηφίο, από δεξιά προς τα αριστερά. Η Εικόνα 7 παρουσιάζει ένα παράδειγμα με την πορεία που ακολουθεί ένα μήνυμα από τον κόμβο με ID=67493 στον κόμβο με ID=34567. Τα ψηφία κινούνται από δεξιά προς τα αριστερό ως εξής:

xxxx7 → xxx67 → xx567 → x4567 → 34567



Εικόνα 7: Tapestry: Ένα παράδειγμα μηχανισμού δρομολόγησης Plaxton mess με δεκαδικά ψηφία μήκους 5.

Το Plaxton mesh χρησιμοποιεί έναν κόμβο ως ρίζα (*root node*) για κάθε αντικείμενο, ο οποίος χρησιμεύει στο να παρέχει έναν σίγουρο κόμβο από τον οποίο το αντικείμενο μπορεί να εντοπισθεί. Όταν ένα αντικείμενο *a* εισάγεται στο δίκτυο και αποθηκεύεται στον κόμβο n_s, ένας root node n_r ανατίθεται σε αυτό με τη χρήση ενός ντετερμινιστικού αλγορίθμου. Ένα μήνυμα δρομολογείται έπειτα από το n_s στο n_r, αποθηκεύοντας τα δεδομένα υπό μορφή χαρτογράφησης (object id a, storer id n_s) σε όλους τους κόμβους, σύμφωνα με αυτόν τον τρόπο. Κατά τη διάρκεια μιας ερώτησης θέσης, τα μηνύματα που προορίζονται για το a δρομολογούνται αρχικά προς το n_r, έως ότου συναντήσει έναν κόμβο που περιέχει την (a, n_s) αντιστοιχία.

Το Plaxton mesh *προσφέρει*:

- απλό χειρισμό λαθών χάρη στη δυνατότητά του να δρομολογήσει γύρω από ένα απλό σύνδεσμο ή έναν κόμβο με την επιλογή ενός κόμβου με ένα παρόμοια κατάληξη και
- Ικανότητα κλιμάκωσης (με μόνο μειονέκτημα την ύπαρξη των root nodes).
- Οι *περιορισμοί* του περιλαμβάνουν:

- την ανάγκη για τη σφαιρική γνώση που απαιτείται για την ανάθεση και τον προσδιορισμό των κόμβων ρίζας και
- την ευπάθεια των root nodes.

Το Plaxton mesh θεωρεί έναν στατικό πληθυσμό κόμβων. Το Tapestry επεκτείνει το σχεδιασμό του για να προσαρμοστεί στους παροδικούς πληθυσμούς των peer-to-peer δικτύων και να παρέχει προσαρμοστικότητα, ανοχή σε σφάλματα καθώς επίσης και διάφορες βελτιστοποιήσεις που περιγράφονται στο [ZKJ2001], όπως για παράδειγμα να ορίζει περισσότερους root nodes σε κάθε αντικείμενο για να εγγυηθεί αξιοπιστία.

Το Tapestry χρησιμοποιείται από διάφορα συστήματα όπως το Oceanstore [KBCEGGWWZ2000, RW2001], το Mnemosyne [HR2002] και το Scan [CKK2000].

1.9 Distributed Hash Tables

Υπάρχουν πολλές διαφορετικές στρατηγικές για την διαχείριση και αναζήτηση των δεδομένων στα peer-to-peer συστήματα. Μια μη αποδοτική λύση είναι ο κατακλυσμός από αιτήσεις σχεδόν όλων των συστημάτων του δικτύου (αυτό χρησιμοποιείται συνήθως στα unstructured peer-to-peer). Σε αυτές τις περιπτώσεις, τα peer-to-peer συστήματα δεν διαθέτουν σωστή ικανότητα κλιμάκωσης, επειδή το δίκτυο υπερφορτώνεται με αιτήσεις.

Μια πιο αποτελεσματική μέθοδος για την αποθήκευση των δεδομένων που εμφανίστηκε τα τελευταία χρόνια είναι τα Distributed Hash Tables (DHTs), όπως το CAN, το Chord. Με τους αποδοτικούς, με ικανότητα κλιμάκωσης και αυτο-οργανωτικούς αλγορίθμους τους για ανάκτηση και διαχείριση των δεδομένων, τα DHTs προσφέρουν σημαντικά πλεονεκτήματα συγκριτικά με τις μη δομημένες μεθόδους [SW2004]. Τα Distributed Hash Tables σχεδιάστηκαν για τα δομημένα peer-to-peer συστήματα.

Η βασική αρχή ενός DHT είναι να αντιστοιχεί τα δεδομένα σε ένα γραμμικό χώρο διευθύνσεων (address space) και να διανέμει τα κλειδιά και τα αντίστοιχα δεδομένα τους, με ένα δομημένο τρόπο, μεταξύ των κόμβων του δικτύου. Το εύρος των τιμών των hash συναρτήσεων σχηματίζει τον χώρο διευθύνσεων (address space) του DHT. Γενικά, αυτός ο χώρος (space) χωρίζεται σε διαστήματα, τα οποία αντιστοιχίζονται σε ατομικούς κόμβους. Κάθε κόμβος είναι πλέον υπεύθυνος για την διαχείριση των δεδομένων που υπάρχουν στο διάστημα που ελέγχει. Έτσι, το DHT σχηματίζει ένα κατανεμημένο hash πίνακα μεταξύ των κόμβων του δικτύου.

Οι αιτήσεις για δεδομένα ενός διαστήματος πρέπει να απαντηθούν από τον κόμβο που είναι υπεύθυνος για αυτό το διάστημα. Εάν μια αίτηση για ένα άλλο διάστημα φτάσει σε έναν κόμβο, αυτός πρέπει να τη προωθήσει άμεσα στο κατάλληλο διάστημα σύμφωνα με τις πληροφορίες δρομολόγησης του DHT [BKK et al. 2003, HHH et al. 2002]. Επειδή, η τοποθέτηση των δεδομένων γίνεται με δομημένο τρόπο, η αναζήτηση και η διαχείριση

μπορούν να εκτελεστούν γρηγορότερα και πιο αποτελεσματικά από μια μη δομημένη peer-to-peer προσέγγιση. Τα αποτελέσματα μπορούν να βρεθούν έχοντας μικρότερο φόρτο δικτύου, με πολυπλοκότητα της τάξης του $O(\log N)$.

Τα Distributed Hash Tables εκτός από τα σημαντικά πλεονεκτήματα που προσφέρουν, παρουσιάζουν μια αδυναμία: *Η διανομή των δεδομένων μεταξύ των κόμβων που συνεργάζονται*[Simon, Petrak and Wehrle, 2004]. Όλες οι εκτιμήσεις όσον αφορά την πολυπλοκότητα ενός DHT σε σχέση με το κόστος αναζήτησης, το κόστος διαχείρισης και αποθήκευσης, βασίζονται στην υπόθεση ότι τα δεδομένα είναι σχεδόν εξ ίσου μοιρασμένα μεταξύ των κόμβων.

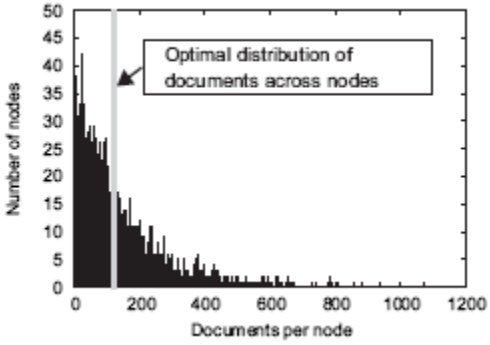
Στις περισσότερες DHT προσεγγίσεις αυτή η υπόθεση βασίζεται στην χρήση των hash συναρτήσεων για την αντιστοίχηση των δεδομένων στον χώρο διεύθυνσης των DHT. Γενικά, υποθέτεται ότι οι hash συναρτήσεις παρέχουν μια διανομή κλειδιών και των αντίστοιχων δεδομένων δια μέσου του DHT χώρου διευθύνσεων. Εάν υπάρχει κάποια σημαντική διαφορά στο φορτίο των κόμβων που αφορούν τα δεδομένα που διαχειρίζεται κάθε κόμβος, το κόστος για την διανομή με αυτό-οργάνωση αυξάνεται δραματικά. Επομένως, απαιτούνται μηχανισμοί εξισορρόπησης φορτίου για να διατηρήσουν την πολυπλοκότητα των DHT αλγορίθμων αναζήτησης στην προτεινόμενη τάξη του $O(\log N)$ ή ακόμα μικρότερη.

Για να αποδειχθεί ότι αυτή η υπόθεση ισχύει και επιπλέον ότι οι μηχανισμοί εξισορρόπησης φορτίου είναι απαραίτητοι για ένα Distributed Hash Table, αναλύεται η κατανομή των φορτίων στα DHTs με ένα πείραμα προσομοίωσης. Κατακερματίζονται (hashed) τα δεδομένα στους χώρους διευθύνσεων ενός Chord δαχτυλιδιού και αναλύεται η διανομή των εγγράφων (documents) μεταξύ των κόμβων (Εικόνα 8). Σε κάθε σενάριο προσομοιώθηκε ένα Distributed Hash Table με 4,096 κόμβους. Οι πολλαπλές προσομοιώσεις που έγιναν για κάθε σενάριο επιβεβαίωσαν τα αποτελέσματα. Ο συνολικός αριθμός των εγγράφων που αποθηκεύτηκαν κυμαινόταν από 100,000 έως 1,000,000. Για αυτό το σκοπό ο χώρος διευθύνσεων του Chord είχε μέγεθος m=22 bits. Συνεπώς, 2^{22} = 4,194,304 documents μπορούσαν να αποθηκευθούν και να διαχειριστούν στο Chord. Τα κλειδιά των δεδομένων και οι κόμβοι δημιουργήθηκαν τυχαία. Στην προσομοίωση αυτή, το φορτίο ενός κόμβου καθορίστηκε από τον αριθμό των αποθηκευμένων documents.

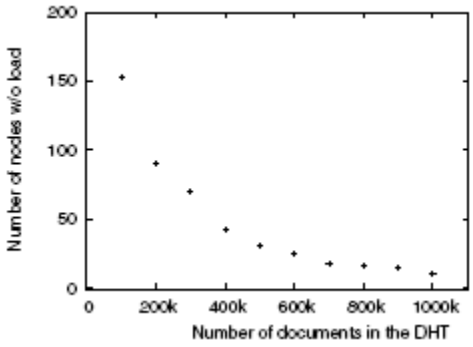
Στην Εικόνα 8 αποδεικνύεται ότι η υπόθεση της ίσης κατανομής των δεδομένων μεταξύ των κόμβων με την απλή χρήση μιας hash συνάρτησης δεν ισχύει. Για παράδειγμα, στην Εικόνα 8 (a) φαίνονται πόσοι κόμβοι (στον άξονα y) αποθηκεύουν έναν σταθερό αριθμό documents (στον άξονα x). Είναι φανερό πως δεν υπάρχει ίση κατανομή documents μεταξύ των κόμβων. Για μια καλύτερη σύγκριση η γκρι γραμμή δείχνει τον κατάλληλο αριθμό στην περίπτωση της ίσης κατανομής, η οποία είναι περίπου 122 documents για κάθε κόμβο.

Επιπλέον, η Εικόνα 8 (b) παρουσιάζει τον αριθμό των κόμβων που δεν έχουν documents.

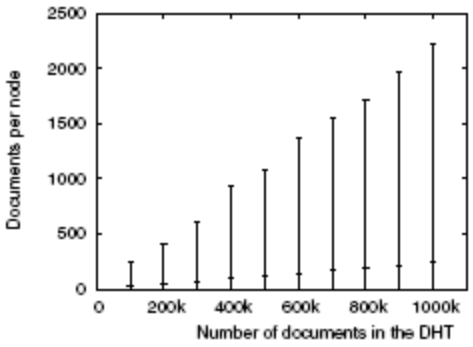
Στην Εικόνα 8 (c) φαίνεται η κατανομή των εγγράφων σε ένα Chord DHT χωρίς εξισορρόπηση φορτίου. Η ποσότητα των εγγράφων που προαναφέρθηκε, κατανεμήθηκε σε 4,096 κόμβους. Η υψηλότερη τιμή δείχνει τον μέγιστο αριθμό των εγγράφων ανά κόμβο και η χαμηλότερη τιμή (συνήθως το 0) τον μικρότερο. Ο κατάλληλος αριθμός των εγγράφων ανά κόμβο σημειώνεται με κουκίδα στο μέσο της στήλης. Ακόμα και για έναν μεγάλο αριθμό εγγράφων σε όλο το DHT, υπάρχουν κάποιοι κόμβοι που δεν διαχειρίζονται κανένα έγγραφο και συνεπώς δεν έχουν φορτίο. Υπάρχουν κόμβοι που διαχειρίζονται δεδομένα με φορτίο 10 φορές μεγαλύτερο από το μέσο όρο.



(a)



(b)



(c)

Εικόνα 8: (a). Η συχνότητα κατανομής του DHT των κόμβων αποθηκεύοντας έναν σταθερό αριθμό documents, (b) Τον αριθμό των κόμβων που δεν έχουν documents, (c) Ο μικρότερος, ο μέσος όρος και ο μεγαλύτερος αριθμός των documents ανά κόμβο.

Συμπερασματικά, είναι φανερό πως δεν μπορεί να επιτευχθεί ίση κατανομή των δεδομένων με τη χρήση μιας απλής hash συνάρτησης. Συνεπώς, θα πρέπει να αναπτυχθούν επιπλέον μηχανισμοί για την εξισορρόπηση του φορτίου δεδομένων μεταξύ των κόμβων. Στις επόμενες ενότητες παρουσιάζονται και αναπτύσσονται μηχανισμοί (αλγόριθμοι) που βοηθούν στην εξισορρόπηση φορτίου στα DHT συστήματα.

Πριν την ανάπτυξη αυτών των αλγορίθμων, πρέπει να προσδιοριστεί ο όρος φορτίο ("*load*"). Επίσης, πρέπει να καθοριστεί πότε ένας κόμβος καλείται υπερφορτωμένος ("*overloaded*") και πότε ένα σύστημα είναι κατάλληλα ισορροπημένο.

Συνήθως, το φορτίο των δεδομένων ενός κόμβου καθορίζεται από το σύνολο των αποθηκευμένων δεδομένων ανά κόμβο. Σε αυτές τις ενότητες, το φορτίο ενός κόμβου θεωρείται το άθροισμα όλων των αποθηκευμένων δεδομένων σε ένα κόμβο. Το συνολικό φορτίο των δεδομένων ενός Peer-to-Peer συστήματος ορίζεται ως το άθροισμα των φορτίων όλων των κόμβων του συστήματος. Το φορτίο ενός συστήματος με *N* κόμβους θεωρείται εξισορροπημένο, αν το φορτίο των δεδομένων κάθε κόμβου είναι το *1/N* του συνολικού φορτίου. Επομένως, ένας κόμβος χαρακτηρίζεται "overloaded ή heavy", εάν το φορτίο του υπερβαίνει το 1/N του συνολικού φορτίου και "light" όταν το φορτίο του είναι μικρότερο από αυτή τη τιμή.

Chord

Στην ενότητα αυτή αναλύουμε το πώς γίνεται η εξισορρόπηση φορτίου στο Chord, στο οποίο αναφερθήκαμε σε προηγούμενη ενότητα.

Το *πρωτόκολλο Chord* υποστηρίζει μια και μόνο λειτουργία: *γνωρίζοντας το κλειδί, το αντιστοιχεί σε έναν κόμβο* [Stoica et al., 2001]. Ανάλογα με την εφαρμογή που χρησιμοποιεί το Chord, ο κόμβος είναι αρμόδιος για την αποθήκευση μιας τιμής που

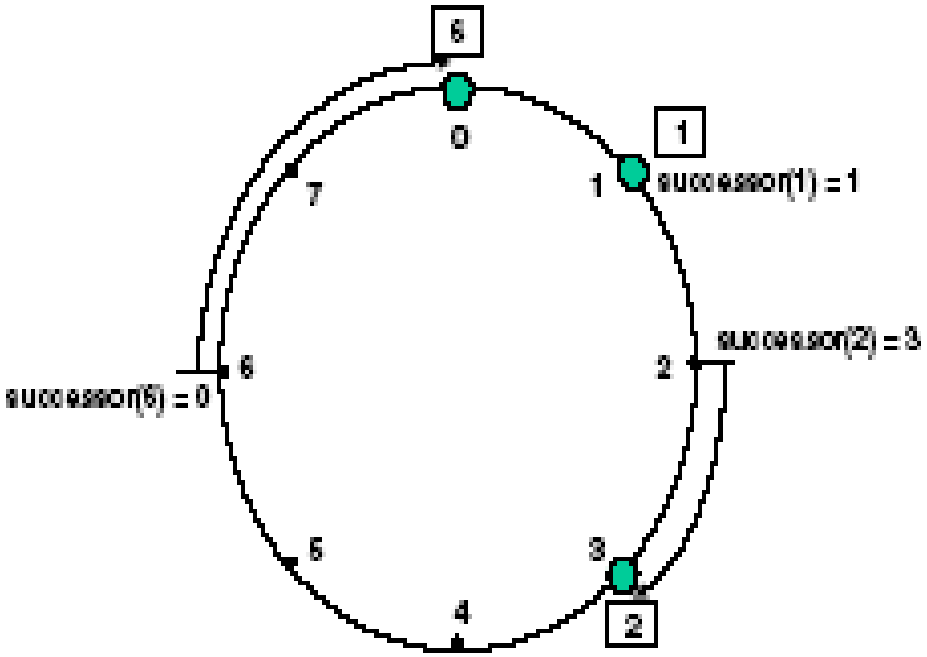
είναι σχετική με το κλειδί. Το Chord χρησιμοποιεί μια παραλλαγή της consistent hashing [KLLLLP1997] για να ορίσει τα κλειδιά στους κόμβους. Η consistent hashing χρησιμοποιείται για την εξισορρόπηση του φορτίου, δεδομένου ότι κάθε κόμβος λαμβάνει κατά προσέγγιση τον ίδιο αριθμό κλειδιών, και περιλαμβάνει σχετικά λίγη μετακίνηση των κλειδιών όταν οι κόμβοι συνδέονται και αποσυνδέονται από το σύστημα.

Τρία χαρακτηριστικά γνωρίσματα που κάνουν το Chord ξεχωρίζει από τα άλλα peer-to-peer πρωτόκολλα αναζήτησης είναι η απλότητα, η αποδείξιμη ακρίβεια (provable correctness) και η αποδείξιμη απόδοσή του. Το Chord είναι απλό, δρομολογώντας ένα κλειδί μέσω μιας ακολουθίας από O(log N) άλλων κόμβων προς τον προορισμό. Ένας κόμβος επιθυμεί πληροφορίες για O(log N) άλλους κόμβους για να εγγυάται αποδοτική δρομολόγηση, αλλά η απόδοση μειώνεται όταν οι πληροφορίες δρομολόγησης είναι μη συνεπείς. Αυτό είναι σημαντικό πρακτικά επειδή οι κόμβοι συνδέονται και αποσυνδέονται αυθαίρετα. Μόνο ένα κομμάτι πληροφοριών ανά κόμβο χρειάζονται να είναι σωστές με σκοπό το Chord να εγγυάται σωστή (αν και αργή) δρομολόγηση των ερωτήσεων. Το Chord έχει έναν απλό αλγόριθμο για αυτές τις πληροφορίες σε ένα δυναμικό περιβάλλον.

Η consistent hash συνάρτηση αναθέτει σε κάθε κόμβο και κλειδί έναν identifier από m bits χρησιμοποιώντας μια βασική hash συνάρτηση, όπως την SHA-1 [FIPS1995]. Ο identifier ενός κόμβου προσδιορίζεται χρησιμοποιώντας τη συνάρτηση hash πάνω στη διεύθυνση IP του κόμβου, ενώ ο identifier ενός κλειδιού παράγεται με την ίδια πράξη πάνω στο κλειδί. Το μήκος του identifier θα πρέπει να είναι αρκετά μεγάλο ώστε η πιθανότητα δύο κόμβοι ή κλειδιά να πάρουν την ίδια τιμή hash για τον identifier να είναι αμελητέα.

Το consistent hashing αναθέτει τα κλειδιά στους κόμβους ως εξής: οι identifiers τοποθετούνται σε έναν κύκλο (identifier circle) modulo 2^m. Το κλειδί *k* ανατίθεται στον πρώτο κόμβο του οποίου ο identifier είναι ίσος ή ακολουθεί το k στο χώρο που ορίζουν οι identifiers. Αυτός ο κόμβος καλείται *successor* του κλειδιού k, και σημειώνεται ως από το *successor(k)*. Εάν οι identifiers τοποθετούνται σε έναν κύκλο από αριθμούς από 0 έως 2^m-1, τότε ο successor(k) είναι πρώτος κόμβος από το k καθώς κινείται από τα δεξιά προς τα αριστερά.

Η Εικόνα 9 παρουσιάζει ένα κύκλο με m=3. Ο κύκλος έχει τρεις κόμβους: 0, 1 και 3. Ο successor του identifier 1 είναι ο κόμβος 1, έτσι το κλειδί 1 τοποθετείται στον κόμβο 1.Ομοίως, το κλειδί 2 τοποθετείται στον κόμβο 3, και το κλειδί 6 στον κόμβο 0.



Εικόνα 9: Ένας identifier circle που περιέχει τρεις κόμβους 0,1 και 3.

Το consistent hashing έχει σχεδιαστεί για να μπορούν οι κόμβοι να συνδέονται και να αποσυνδέονται από το δίκτυο χωρίς προβλήματα. Για να διατηρηθεί το consistent hashing, όταν ένας κόμβος n συνδεθεί στο δίκτυο, ορισμένα κλειδιά που βρίσκονταν προηγουμένως στον successor του *n* τώρα δίνονται στον *n*. Όταν ένας κόμβος *n* αποσυνδεθεί από το δίκτυο, τα κλειδιά του αναθέτονται στον successor του. Καμία άλλη αλλαγή στην ανάθεση των κλειδιών στους κόμβους δεν εμφανίζεται. Στο παραπάνω παράδειγμα, εάν ένας κόμβος επρόκειτο να συνδεθεί με τον identifier 7, αυτό θα έπαιρνε το κλειδί με το identifier 6 από τον κόμβο με το identifier 0.

ΘΕΩΡΗΜΑ $1^{Ο}$: Για οποιοδήποτε σύνολο N κόμβων και K κλειδιών, με μεγάλη πιθανότητα ισχύει:

- Κάθε κόμβος είναι υπεύθυνος για το πολύ $(1+\epsilon)K/N$ κλειδιά,
- Όταν ένας $(N+1)^{st}$ κόμβος συνδέεται ή αποσυνδέεται από το δίκτυο, η ευθύνη για O(K/N) κλειδιά αλλάζει χέρια

Ένα πολύ μικρό ποσό πληροφοριών δρομολόγησης αρκεί για να εφαρμοστεί μια consistent hashing σε ένα κατανεμημένο περιβάλλον. Κάθε κόμβος το μόνο που χρειάζεται να γνωρίζει είναι τον successor του στον κύκλο. Οι ερωτήσεις για έναν συγκεκριμένο identifier περνούν από τον κύκλο διάμεσου των δεικτών του successor

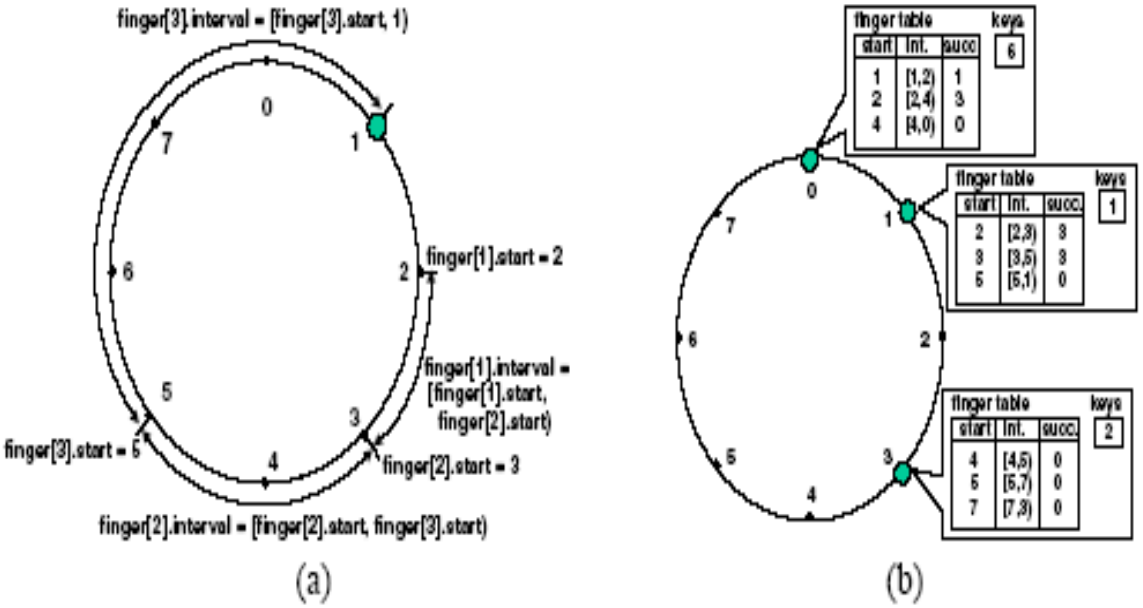
μέχρι να συναντήσουν έναν κόμβο που αντιστοιχεί στον identifier. Αυτός είναι ο κόμβος στον οποίο αντιστοιχεί και η ερώτηση. Ένα τμήμα του Chord πρωτοκόλλου διατηρεί αυτούς τους δείκτες του successor, εξασφαλίζοντας έτσι ότι όλες οι αναζητήσεις γίνονται σωστά. Ωστόσο, αυτή η ανάλυση είναι ανεπαρκής: μπορεί να απαιτηθεί να διατρέξουμε και τους N κόμβους για να βρεθεί η αντιστοίχηση. Για να επιταχύνει αυτήν την διαδικασία, το Chord διατηρεί πρόσθετες πληροφορίες δρομολόγησης. Αυτές οι πρόσθετες πληροφορίες δεν είναι ουσιαστικές για την ορθότητα (correctness), η οποία επιτυγχάνεται εφ' όσον οι πληροφορίες του successor διατηρούνται σωστά.

Κάθε κόμβος, *n*, διατηρεί έναν πίνακα δρομολόγησης με το πολύ *m* καταχωρήσεις (με *m* ο αριθμός των bits του identifier του κλειδιού/ κόμβου). Αυτός ο πίνακας ονομάζεται *finger table*. Η i^{th} είσοδος στον πίνακα στον κόμβο *n* περιέχει την ταυτότητα του *πρώτου* κόμβου, *s*, ο οποίος βρίσκει τον *n* μετά από τουλάχιστον 2^{i-1} πάνω στον κύκλο. Η είσοδος, δηλαδή $s = successor(n + 2^{i-1})$, όπου $1 \le i \le m$ (και όλη η αριθμητική είναι modulo 2^m). Ο κόμβος *s* ονομάζεται το i^{th} *finger* του κόμβου *n*, και δίνεται από το *n.finger[i].node* (Πίνακας 3). Ένα στοιχείο του finger table περιλαμβάνει και τον identifier του Chord και την IP διεύθυνση (και τον αριθμό της θύρας) του αντίστοιχου κόμβου. Σημειώστε ότι το πρώτο finger του *n* είναι ο άμεσος *successor* του στον κύκλο. Για ευκολία αναφέρεται συχνά ως *successor* παρά σαν πρώτος δείκτης (finger).

Πίνακας 3. Ο καθορισμός των μεταβλητών για τον κόμβο n, χρησιμοποιώντας m bits identifiers.

Notation	Definition
$finger[k].start$	$(n + 2^{k-1}) \bmod 2^m,\ 1 \le k \le m$
$.interval$	$[finger[k].start, finger[k+1].start)$
$.node$	first node $\ge n.finger[k].start$
successor	the next node on the identifier circle; $finger[1].node$
predecessor	the previous node on the identifier circle

Στο παράδειγμα της Εικόνα 10 (β), το finger table του κόμβου 1 περιέχει τον successor των κόμβων με identifier $(1 + 2^0) \bmod 2^3 = 2$, $(1 + 2^1) \bmod 2^3 = 3$ και $(1 + 2^2) \bmod 2^3 = 5$, αντίστοιχα. Ο successor του identifier 2 είναι ο κόμβος 3, καθώς αυτός είναι ο πρώτος κόμβος που ακολουθεί τον 2, ο successor του identifier 3 είναι ο κόμβος 3 και ο successor του 5 είναι κόμβος 0.



Εικόνα 10: (a) Τα finger διαστήματα του κόμβου 1. (b) Το finger table και το κλειδί για ένα δίκτυο με κόμβους 0,1 και 3 και κλειδιά 1,2 και 6.

Αυτό το σχήμα έχει δύο σημαντικά χαρακτηριστικά. Πρώτον, κάθε κόμβος αποθηκεύει πληροφορίες για έναν μικρό αριθμό άλλων κόμβων, και γνωρίζει περισσότερα για τους κόμβους που είναι κοντά του πάνω στο κύκλο ταυτοτήτων απ' ότι για αυτούς που είναι μακριά του. Δεύτερον, το finger table ενός κόμβου γενικά δεν περιέχει αρκετές πληροφορίες για να καθορίσει τον successor ενός κλειδιού *k*. Για παράδειγμα, ο κόμβος 3 στην Εικόνα 10 δεν γνωρίζει τον successor του 1, αφού ο successor του 1 δεν εμφανίζεται στον finger table του κόμβου 3.

Τι συμβαίνει όταν ένας κόμβος *n* δεν ξέρει τον successor ενός κλειδιού *k*; Εάν ένας κόμβος *n* μπορεί να βρεί έναν κόμβο του οποίου η ταυτότητα είναι πιο κοντά από το δικό του κλειδί, ο κόμβος αυτός θα γνωρίζει περισσότερα από τον *n* για τον κύκλο ταυτοτήτων στην περιοχή του *k*. Κατά συνέπεια το *n* αναζητά στο finger table τον κόμβο *j* του οποίου η ταυτότητα προηγείται του *k*, και ρωτά το *j* για τον κόμβο που γνωρίζει ποια ταυτότητα είναι πιο κοντά στο *k*. Με την επανάληψη αυτής της διαδικασίας, το *n* μαθαίνει τους κόμβους που είναι πιο κοντά στην ταυτότητά του και πιο κοντά στο *k*.

ΘΕΩΡΗΜΑ 2^{Ο}: Με μεγάλη πιθανότητα, ο αριθμός των κόμβων, ο οποίος πρέπει να συνδεθεί για να βρεθεί ένας successor μέσα σε ένα δίκτυο με N κόμβους είναι $O(\log N)$.

ΘΕΩΡΗΜΑ 3^{Ο}: Με μεγάλη πιθανότητα, κάθε συνδεόμενος ή αποσυνδεόμενος κόμβος σε ένα Chord δίκτυο, με N κόμβους, θα χρησιμοποιεί μηνύματα της τάξης του $O(\log_2 N)$, για να επανεγκαταστήσει τις σταθερές δρομολόγησης και τα finger tables του Chord.

Για να απλοποιηθούν οι μηχανισμοί σύνδεσης και αποσύνδεσης, κάθε κόμβος στο Chord διατηρεί έναν *predecessor pointer*. Ο predecessor pointer ενός κόμβου περιέχει τον identifier του Chord και την IP address του αμέσως επόμενου predecessor του συγκεκριμένου κόμβου, και μπορεί να χρησιμοποιηθεί για να κινηθεί με φορά αντίθετης των δεικτών ενός ρολογιού πάνω στον κύκλο του identifier.

Για να διατηρηθούν οι σταθερές που προαναφέρθηκαν, το Chord πρέπει να πραγματοποιήσει τρεις ενέργειες όταν ένας *n* κόμβος συνδεθεί με το δίκτυο:

1. Να οριστούν ο predecessor και τα fingers του κόμβου *n*.
2. Να ενημερωθούν τα fingers και οι predecessors των υπαρχόντων κόμβων για να προσθήκη του *n* στο δίκτυο.
3. Να πληροφορηθεί το λογισμικό ανώτερου επιπέδου, ώστε να μπορεί να μεταφέρει αντικείμενα (για παράδειγμα τιμές) που σχετίζονται με τα κλειδιά για τα οποία ο κόμβος *n* είναι πια υπεύθυνος.

2. Υπολογισμός σε Νέφος (Cloud Computing)

Ο υπολογισμός σε νέφος είναι όνειρο πολλών ετών στον υπολογισμό, έχει τη δυνατότητα να μετατρέψει ένα μεγάλο κομμάτι της βιομηχανίας της Τεχνολογίας της Πληροφορίας, κάνοντας το λογισμικό πολύ πιο ελκυστικό σαν υπηρεσία και μορφοποιώντας τον τρόπο που το υλικό σχεδιάζεται και πωλείται (δες το AFGJKKLPRSZ2009). Κατασκευαστές λογισμικού με πρωτοπόρες ιδέες για νέες υπηρεσίες πάνω από το Internet δεν χρειάζονται πλέον τις νέες μεγάλες ανακαλύψεις στο υλικό για να αναπτύξουν τις υπηρεσίες τους ή δαπάνες για να τις κάνουν να λειτουργήσουν. Δεν χρειάζεται πλέον να ανησυχούν για υπερ-παροχή υπηρεσίας της οποίας η δημοφιλία δεν έφτασε τις προβλέψεις τους ή υπο-παροχή υπηρεσίας που έγινε πολύ διάσημη χάνοντας έτσι πελάτες και έσοδα. Επιπλέον, οι εταιρείες με μεγάλα batch-oriented έργα μπορούν να πάρουν αποτελέσματα τόσο σύντομα όσο το επιτρέπει η κλιμάκωση των προγραμμάτων τους, αφού η χρήση 1000 εξυπηρετητών για 1 ώρα δεν κοστίζει περισσότερο από τη χρήση ενός εξυπηρετητή για 1000 ώρες. Αυτή η ελαστικότητα των πόρων, χωρίς την πληρωμή έξτρα χρημάτων για μεγάλη κλιμάκωση δεν έχει προηγούμενο στην ιστορία της τεχνολογίας της πληροφορίας.

Ο υπολογισμός σε νέφος αναφέρεται τόσο στις εφαρμογές που διατίθενται σαν υπηρεσίες πάνω από το Internet και στο υλικό και τα συστήματα λογισμικού στα Κέντρα Αποθήκευσης Δεδομένων (Data Centers) που παρέχουν τις υπηρεσίες αυτές. Οι ίδιες οι υπηρεσίες εδώ και καιρό αναφέρονται σαν Λογισμικό σαν Υπηρεσία (Software as a Service – SaaS). Το υλικό και το λογισμικό ενός Κέντρου Αποθήκευσης Δεδομένων είναι αυτό που ονομάζουμε Νέφος (Cloud). Όταν ένα νέφος γίνεται διαθέσιμο στο ευρύ κοινό με ένα τρόπο πληρωμής με βάση τη χρήση λέγεται Δημόσιο Νέφος (Public Cloud) και η υπηρεσία που πωλείται ονομάζεται Utility Computing. Ο όρος Ιδιωτικό Νέφος (Private Cloud) για να αναφερθούμε σε εσωτερικά Data Centers μιας επιχείρησης ή ενός οργανισμού που δεν είναι διαθέσιμα στο ευρύ κοινό. Άρα ο υπολογισμός σε νέφος είναι το άθροισμα του SaaS και του Utility Computing, αλλά δεν περιλαμβάνει ιδιωτικά νέφη. Οι άνθρωποι μπορεί να είναι είτε χρήστες ή πάροχοι των SaaS είτε χρήστες ή πάροχοι μιας Utility Computing.

Από την άποψη του υλικού τρία είναι τα νέα στοιχεία στον υπολογισμό νέφους:

1 Η αίσθηση των άπειρων υπολογιστικών πόρων που είναι διαθέσιμοι κατ' απαίτηση, περιορίζοντας την ανάγκη των χρηστών σχεδιάσουν το πώς θα γίνει η παροχή των πόρων.

2 Ο περιορισμός μιας από πάνω προς τα κάτω δέσμευσης των χρηστών του νέφους, που επιτρέπει στις εταιρείες να ξεκινήσουν με λίγους πόρους που θα μπορούν να τους αυξήσουν στη συνέχεια αναλόγων της αύξησης των απαιτήσεων.

3 Η δυνατότητα να πληρώνεις για τους υπολογιστικούς πόρους σε βραχυπρόθεσμη βάση αναλόγως των αναγκών (π.χ. επεξεργαστές που πληρώνονται με την ώρα και αποθήκευση με την ημέρα) και απελευθέρωσή τους σύμφωνα με τις ανάγκες.

Θεωρούμε ότι η δημιουργία και η λειτουργία εξαιρετικά μεγάλης κλίμακας κέντρων αποθήκευσης δεδομένων σε θέσεις χαμηλού κόστους, ήταν το κλειδί για την έκρηξη του Cloud Computing. Όταν αναφερόμαστε σε χαμηλό κόστος αναφερόμαστε σε κόστος κατανάλωσης ρεύματος, εύρους ζώνης, λειτουργιών, λογισμικού και υλικού. Όλοι αυτοί οι παράγοντες αν συνδυαστούν οδηγούν στο συμπέρασμα ότι ο υπολογισμός σε νέφος μπορεί να προσφέρει υπηρεσίες κάτω από το κόστος ενός μεσαίου μεγέθους κέντρου αποθήκευσης δεδομένων φέρνοντας έτσι καλό κέρδος.

Κάθε εφαρμογή απαιτεί ένα μοντέλο υπολογισμού, ένα μοντέλο αποθήκευσης και ένα μοντέλο επικοινωνίας. Η στατιστική πολυπλεξία που είναι απαραίτητη για να επιτευχθεί η ελαστικότητα και η αίσθηση της απεριόριστης χωρητικότητας απαιτεί κάθε ένας από τους πόρους αυτούς να γίνει εικονικός ώστε να κρύβει την υλοποίηση του πως πολυπλέκονται και διαμοιράζονται.

Το Amazon EC2 βρίσκεται στη μία άκρη του χώρου αυτού. Ένα EC2 στιγμιότυπο μοιάζει περισσότερο με φυσικό υλικό και οι χρήστες μπορούν να ελέγχουν ολόκληρη τη στοίβα λογισμικού, από τον πυρήνα προς τα πάνω. Το χαμηλό αυτό επίπεδο κάνει δύσκολο για την Amazon να προσφέρει αυτόματη διαβαθμισιμότητα και ανάκαμψη από σφάλματα, γιατί η σημασιολογία που συσχετίζεται με την αναπαραγωγή αντιγράφων (replication) και άλλα θέματα διαχείρισης κατάστασης είναι σε μεγάλο βαθμό εξαρτώμενα από την εφαρμογή. Στην άλλη άκρη βρίσκεται συγκεκριμένες πλατφόρμες πεδίου εφαρμογών όπως το Google AppEngine. Το AppEngine στοχεύει αποκλειστικά σε παραδοσιακές εφαρμογές ιστού, θεωρώντας μια δομή εφαρμογής καθαρού διαχωρισμού ανάμεσα στο επίπεδο stateless υπολογισμού και το επίπεδο stateful αποθήκευσης. Η εντυπωσιακή αυτόματη διαβαθμισιμότητα και οι μηχανισμοί υψηλής διαθεσιμότητας του AppEngine και το ιδιωτικό MegaStore αποθήκευσης δεδομένων που είναι διαθέσιμη στις AppEngine εφαρμογές, βασίζονται σε αυτούς τους περιορισμούς. Οι εφαρμογές για το Azure της Microsoft γράφονται χρησιμοποιώντας .NET βιβλιοθήκες και μεταφράζονται σε Common Language Runtime, ένα περιβάλλον διαχείρισης ανεξάρτητο από κάποια γλώσσα. Έτσι το Azure βρίσκεται ανάμεσα πλαίσια εφαρμογών όπως το AppEngine και εικονικές μηχανές υλικού όπως το EC2.

Πότε όμως μια Utility Computing είναι προτιμότερη από το Ιδιωτικό Νέφος; Αρχικά όταν η ζήτηση για υπηρεσίες ποικίλει με το χρόνο. Η παροχή ενός κέντρου αποθήκευσης δεδομένων για φόρτο σε ώρες αιχμής μπορεί να διατηρηθεί για λίγες μέρες σε ένα μήνα, οδηγώντας σε υποχρησιμοποίηση σε άλλες χρονικές περιόδους για παράδειγμα. Αντίθετα, ο υπολογισμός σε νέφος επιτρέπει σε ένα οργανισμό να πληρώνει με την ώρα για υπολογιστικούς πόρους, οδηγώντας πιθανότητα σε κέρδος ακόμη κι αν το ωριαίο κόστος της ενοικίασης μιας μηχανής από ένα πάροχο είναι υψηλότερο από το να κατέχεις

μία μηχανή. Επίσης, μια δεύτερη περίπτωση είναι όταν η ζήτηση είναι άγνωστη εκ των προτέρων. Τέλος, οργανισμοί που εκτελούν ομαδικές (batch) αναλύσεις μπορούν να χρησιμοποιήσουν τη «σχέση κόστους» του υπολογισμού νέφους για να ολοκληρώσουν υπολογισμούς γρηγορότερα: χρησιμοποιώντας 1000 EC2 μηχανές για 1 ώρα κοστίζει το ίδιο με τη χρήση μιας μηχανής για 1000 ώρες. Για την πρώτη περίπτωση μιας επιχείρησης υπηρεσιών ιστού με ποικίλει ζήτηση με βάση το χρόνο και εισόδημα (revenue) ανάλογο των ωρών των χρηστών, δίνουμε τη σχέση με την παρακάτω εξίσωση:

$$\text{UserHours}_{\text{cloud}} \times (\text{revenue} - \text{Cost}_{\text{cloud}}) \geq \text{UserHours}_{\text{cloud}} \times (\text{revenue} - (\text{Cost}_{\text{datacenter}}/\text{Utilization}))$$

Το αριστερό μέλος της εξίσωσης πολλαπλασιάζει το καθαρό εισόδημα ανά ώρα χρήστη, δίνοντας το εκτιμώμενο κέρδος από τη χρήση του υπολογιστικού νέφους. Το δεξιό μέλος εκτελεί τον ίδιο υπολογισμό για συγκεκριμένης χωρητικότητας κέντρο αποθήκευσης δεδομένων χρησιμοποιώντας τη μέση χρησιμοποίηση, μη συμπεριλαμβανομένων των φόρτων αιχμής στο κέντρο αποθήκευσης δεδομένων.

2.1 Τεχνολογίες

Υπάρχουν συγκεκριμένες τεχνολογίες που λειτουργούν πίσω από τις πλατφόρμες υπολογιστικού νέφους που κάνουν ένα νέφος ευέλικτο, αξιόπιστο και χρήσιμο (δες το [cloud]). Οι τεχνολογίες αυτές είναι οι εξής:

- Εικονικοποίηση (Virtualization)
- Αρχιτεκτονική προσανατολισμένη στην υπηρεσία (service – oriented architecture)
- Υπολογισμός σε πλέγμα (Grid Computing)
- Utility Computing

Εικονικοποίηση (Virtualization)

Τρεις θεμελιώδεις κλάσεις αφαίρεσης – διερμηνευτές, μνήμη και σύνδεσμοι επικοινωνίας – είναι απαραίτητες για την περιγραφή της λειτουργίας ενός υπολογιστικού συστήματος. Η φυσική υλοποίηση καθεμιάς από τις τρεις τους, όπως οι επεξεργαστές που μετασχηματίζουν την πληροφορία, πρωτεύουσα και δευτερεύουσα μνήμη για την αποθήκευση της πληροφορίας, και κανάλια επικοινωνίας που επιτρέπουν διαφορετικά συστήματα να επικοινωνούν μεταξύ τους, μπορούν να ποικίλουν σε εύρος ζώνης με την ευρεία έννοια σημαίνει τον αριθμό των πράξεων ανά μονάδα χρόνου), καθυστέρηση (ο χρόνος από την στιγμή που μια λειτουργία αρχικοποιείται μέχρι να αρχίσουν να γίνονται ορατά τα αποτελέσματά της), αξιοπιστία και άλλα φυσικά χαρακτηριστικά. Συστήματα λογισμικού, όπως τα λειτουργικά συστήματα είναι υπεύθυνα για τη διαχείριση των πόρων του συστήματος, τις φυσικές υλοποιήσεις των τριών παραπάνω κλάσεων [M2013].

Η διαχείριση πόρων γίνεται πολύπλοκη, καθώς η κλίμακα του συστήματος όπως και ο αριθμός των χρηστών και η ποικιλία των εφαρμογών που χρησιμοποιούν το σύστημα αυξάνει. Η διαχείριση πόρων για μια κοινότητα χρηστών με ευρεία έκταση εφαρμογών που εκτελούνται υπό την επίβλεψη ενός λειτουργικού συστήματος είναι ένα πολύ δύσκολο πρόβλημα. Επιπλέον των εξωτερικών παραγόντων, η διαχείριση των πόρων επηρεάζεται από εσωτερικούς παράγοντες, όπως η ετερογένεια του υλικού και των συστημάτων λογισμικού, η ικανότητα προσέγγισης της σφαιρικής κατάστασης του συστήματος και η ανακατανομή του φόρτου, ο ρυθμός σφαλμάτων των διαφόρων συστατικών και πολλοί άλλοι παράγοντες.

Η παραδοσιακή λύση για ένα κέντρο αποθήκευσης δεδομένων (data center) είναι η εγκατάσταση λειτουργικών συστημάτων σε αυτόνομα συστήματα και η στήριξη σε τεχνικές λειτουργικών συστημάτων για την επιβεβαίωση της διαμοίρασης πόρων, προστασίας εφαρμογών, και την απομόνωση της απόδοσης. Η διαχείριση συστημάτων, η καταγραφή των δραστηριοτήτων, η ασφάλεια και η διαχείριση των πόρων είναι μεγάλες προκλήσεις για τους παρόχους υπηρεσιών με βάση αυτή τη λογική. Η ανάπτυξη εφαρμογών και η βελτιστοποίηση εφαρμογών είναι μεγάλες προκλήσεις για τους χρήστες.

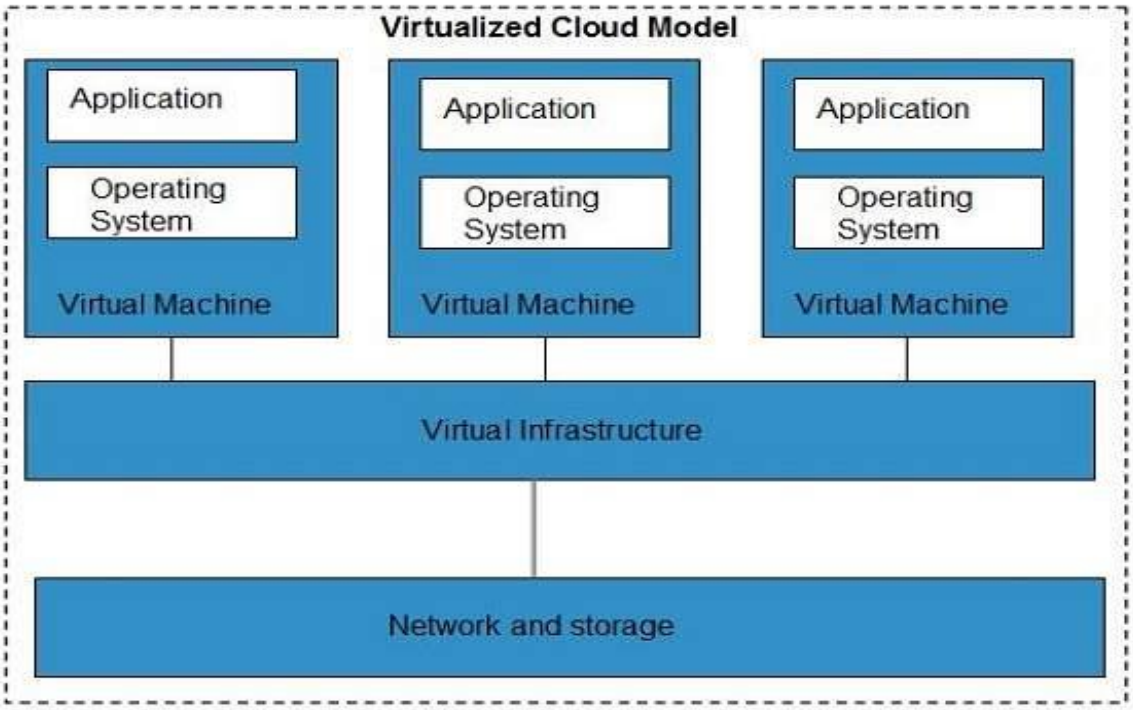
Η εναλλακτική λύση είναι η εικονικοποίηση των πόρων. Η εικονικοποίηση απλοποιεί κάποιες από τις εργασίες διαχείρισης πόρων. Η εικονικοποίηση είναι μια τεχνολογία που επιτρέπει τη διαμοίραση απλών στιγμιότυπων μιας εφαρμογής ή ενός πόρου μεταξύ πολλαπλών οργανισμών ή πελατών. Το επιτυγχάνει αναθέτοντας ένα λογικό όνομα σε ένα φυσικό πόρο και παρέχοντας ένα δείκτη προς το φυσικό αυτό πόρο όταν ζητείται. Για παράδειγμα, η κατάσταση μιας εικονικής μηχανής (virtual machine – VM) που εκτελείται υπό την επίβλεψη ενός παρακολουθητή εικονικών μηχανών (virtual machine monitor – VMM) μπορεί να αποθηκευτεί και να μετακινηθεί σε άλλον εξυπηρετητή για την εξισορρόπηση του φόρτου. Την ίδια χρονική στιγμή, η εικονικοποίηση επιτρέπει στους χρήστες να λειτουργούν σε περιβάλλοντα τα οποία τους είναι οικεία.

Η αρχιτεκτονική πολλαπλών πελατών (multitenant architecture) προσφέρει εικονική απομόνωση μεταξύ των πολλαπλών πελατών. Με τον τρόπο αυτό, οι διάφοροι οργανισμοί μπορούν να χρησιμοποιήσουν και να προσαρμόσουν την εφαρμογή τους σαν να έχει ο καθένας τα στιγμιότυπά του σε κατάσταση εκτέλεσης (Εικόνα 11).

Η εικονικοποίηση προσομοιώνει τη διεπαφή με τα φυσικά αντικείμενα με έναν από τους παρακάτω τρόπους:

- Πολύπλεξη. Δημιουργία πολλαπλών εικονικών αντικειμένων ενός στιγμιοτύπου φυσικού αντικειμένου. Για παράδειγμα, ένας επεξεργαστής πολυπλέκεται ανάμεσα σε ένα αριθμό διεργασιών ή νημάτων.

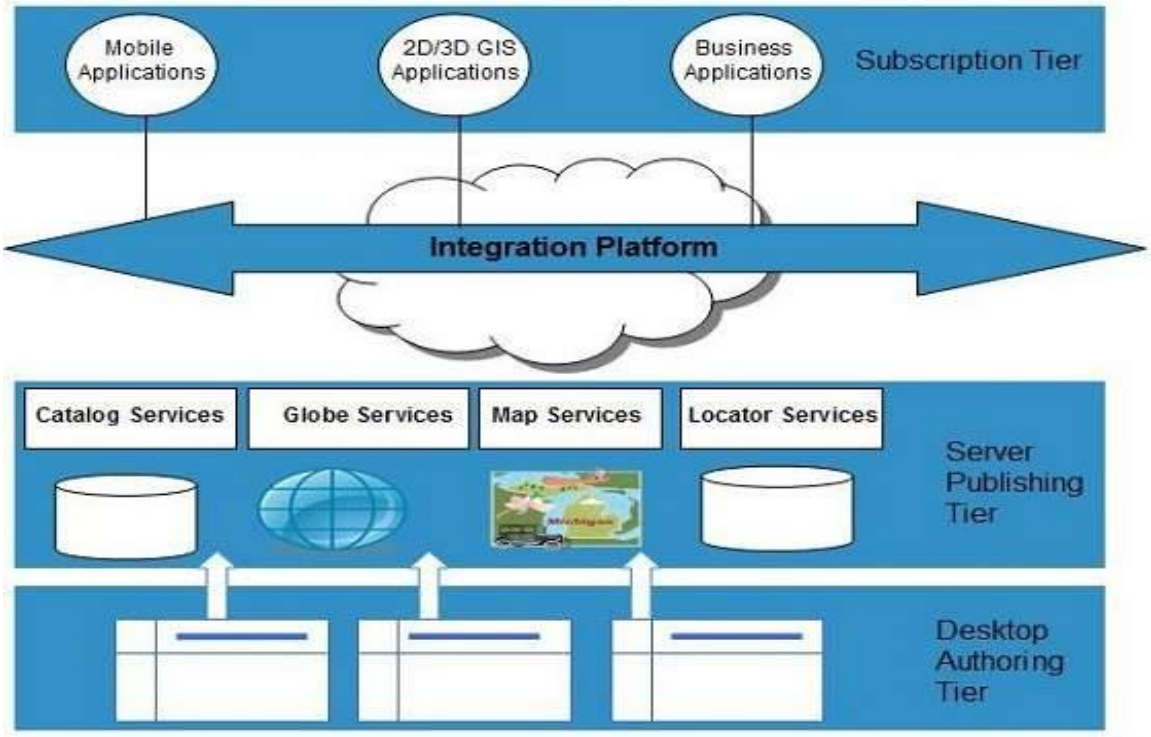
- Συσσωμάτωση. Δημιουργία ενός εικονικού αντικειμένου από πολλαπλά φυσικά αντικείμενα. Για παράδειγμα, ένας αριθμός φυσικών δίσκων ενσωματώνονται σε ένα δίσκο τύπου RAID.
- Εξομοίωση. Κατασκευή ενός εικονικού αντικειμένου από ένα διαφορετικού τύπου φυσικό αντικείμενο. Για παράδειγμα, ένας φυσικός δίσκος που εξομοιώνει την τυχαία πρόσβαση μνήμης.
- Πολύπλεξη και εξομοίωση. Παράδειγμα: εικονική μνήμη με πολύπλεξη σελιδοποίησης πραγματικής μνήμης και δίσκου, και εικονική διεύθυνση που εξομοιώνει μια πραγματική διεύθυνση.



Εικόνα 11: Εικονικοποίηση.

Αρχιτεκτονική προσανατολισμένη στην υπηρεσία (service – oriented architecture)

Η αρχιτεκτονική προσανατολισμένη στην υπηρεσία βοηθά στην χρήση εφαρμογών σαν υπηρεσίες από άλλες εφαρμογές ανεξάρτητα από τον τύπο του κατασκευαστή, του προϊόντος ή της τεχνολογίας. Έτσι είναι δυνατή η ανταλλαγή δεδομένων μεταξύ εφαρμογών διαφορετικών κατασκευαστών χωρίς επιπλέον προγραμματισμό ή αλλαγών σε υπηρεσίες (Εικόνα 12).

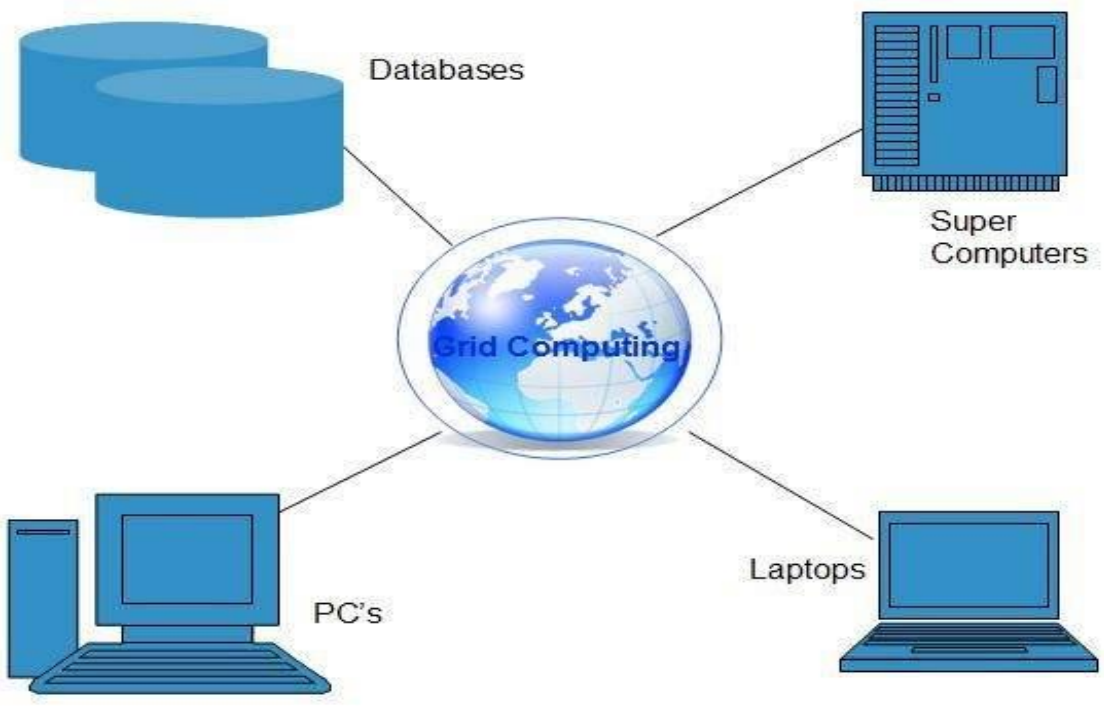


Εικόνα 12: Αρχιτεκτονική προσανατολισμένη στην υπηρεσία.

Υπολογισμός σε πλέγμα (Grid Computing)

Ο υπολογισμός σε πλέγμα αναφέρεται σε κατανεμημένο υπολογισμό, στον οποίο μια ομάδα από υπολογιστές από πολλαπλές τοποθεσίες συνδέονται για να εκπληρώσουν ένα κοινό στόχο. Αυτοί οι υπολογιστικοί πόροι είναι ετερογενείς και γεωγραφικά διασκορπισμένοι.

Ο υπολογισμός σε πλέγμα διαμερίζει τις πολύπλοκες εργασίες σε μικρότερα κομμάτια τα οποία κατανέμονται στις ΚΜΕ που ανήκουν στο πλέγμα (Εικόνα 13).



Εικόνα 13: Υπολογισμός σε πλέγμα.

Utility computing

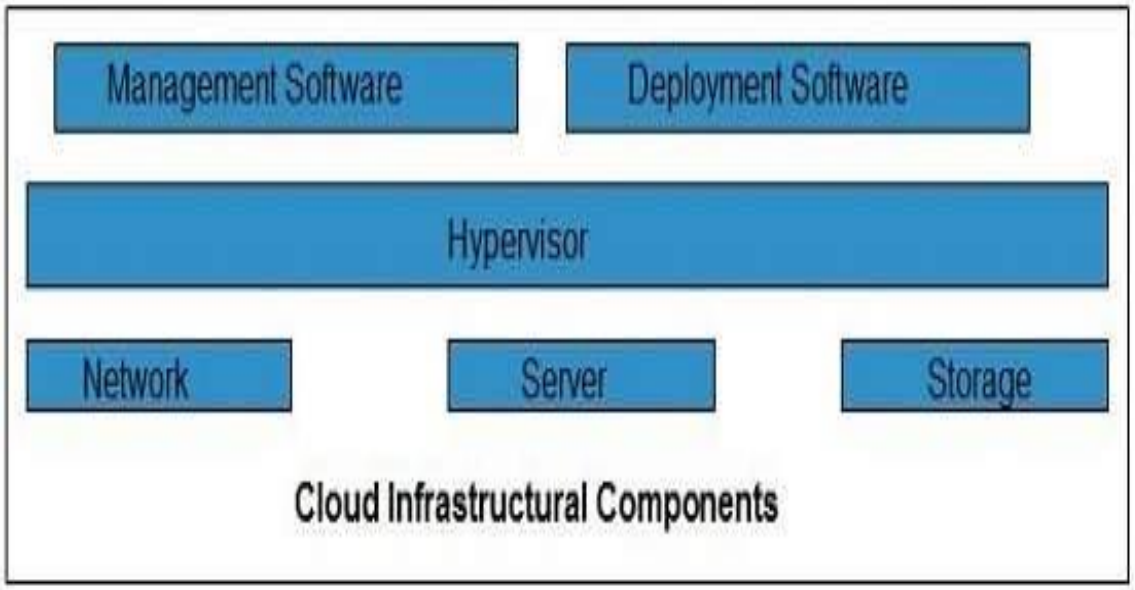
Ένα utility computing βασίζεται σε ένα μοντέλο πληρωμής με βάση τη χρησιμοποίηση (pay per use model). Προσφέρει υπολογιστικούς πόρους κατ' απαίτηση σαν μετρήσιμη υπηρεσία. Ο υπολογισμός σε νέφος, ο υπολογισμός σε πλέγμα και οι διαχειρίσιμες IT υπηρεσίες βασίζονται στην έννοια του utility computing.

2.2 Υποδομές

Η υποδομή ενός νέφους αποτελείται από εξυπηρετητές, συσκευές αποθήκευσης, δίκτυο, λογισμικό διαχείρισης, λογισμικό ανάπτυξης και εικονικοποίηση πλατφόρμας (Εικόνα 14).

Το hypervisor είναι ένα πρόγραμμα χαμηλού επιπέδου (firmware) που δρα σαν διαχειριστής εικονικής μηχανής (VMM). Επιτρέπει τη διαμοίραση απλών φυσικών στιγμιοτύπων ενός πόρου μεταξύ πολλών πελατών. Το λογισμικό διαχείρισης βοηθά στη συντήρηση και τη διαμόρφωση της υποδομής. Το λογισμικό ανάπτυξης βοηθά στην ανάπτυξη και την ενσωμάτωση της εφαρμογής στο νέφος. Το δίκτυο επιτρέπει τη σύνδεση των υπηρεσιών νέφους πάνω από το Internet. Ακόμη, είναι η δυνατή η διαμοίραση δικτύου πάνω από το Internet, πράγμα που σημαίνει ότι ο πελάτης μπορεί να προσαρμόσει το δρομολόγιο δικτύου και το πρωτόκολλο στις ανάγκες του. Ο

εξυπηρετητής βοηθά στη διαμοίραση των πόρων και προσφέρει και άλλες υπηρεσίες όπως παρακολούθηση των πόρων, ασφάλεια, κλπ. Το νέφος διατηρεί επίσης, πολλαπλά αντίγραφα κατά την αποθήκευση. Εάν κάποιος από τους αποθηκευτικούς πόρους καταρρεύσει τότε μπορεί να εξαχθεί από ένα άλλο, κάνοντας έτσι τον υπολογισμό νέφους περισσότερο αξιόπιστο.



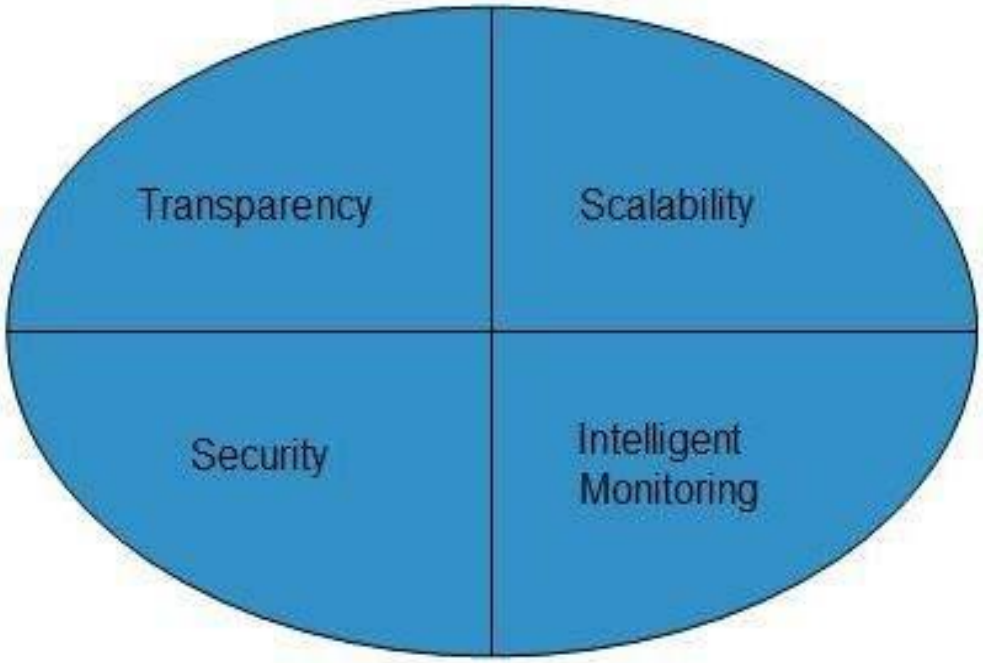
Εικόνα 14: Συστατικά υποδομών νέφους.

Χαρακτηριστικά Υποδομών

Υπάρχουν θεμελιώδη χαρακτηριστικά που μια υποδομή νέφους πρέπει να υλοποιεί, όπως δείχνει και η Εικόνα 15Εικόνα 15: Χαρακτηριστικά υποδομών νέφους..

- Διαφάνεια: Η εικονικοποίηση είναι κλειδί για τη διαμοίραση των πόρων σε περιβάλλοντα νέφους. Αλλά δεν είναι δυνατή η ικανοποίηση της ζήτησης με ένα απλό εξυπηρετητή. Για το λόγο αυτό πρέπει να υπάρχει διαφάνεια στους πόρους, στην εξισορρόπηση φόρτου και στις εφαρμογές, ώστε να μπορεί να μπορεί να γίνει κλιμάκωση σύμφωνα με την ζήτηση.
- Κλιμάκωση: η κλιμάκωση μιας εφαρμογής που διανέμεται δεν είναι το ίδιο εύκολη με την κλιμάκωση μιας απλής εφαρμογής γιατί περιλαμβάνει επιβάρυνση διαμόρφωσης ή επανασχεδίαση της αρχιτεκτονικής του δικτύου. Έτσι, μια εφαρμογή που διανέμεται πρέπει να είναι κλιμακούμενη, το οποίο απαιτεί η υποδομή της εικονικοποίησης να μπορεί να δεσμεύει και να αποδεσμεύει τους πόρους εύκολα.

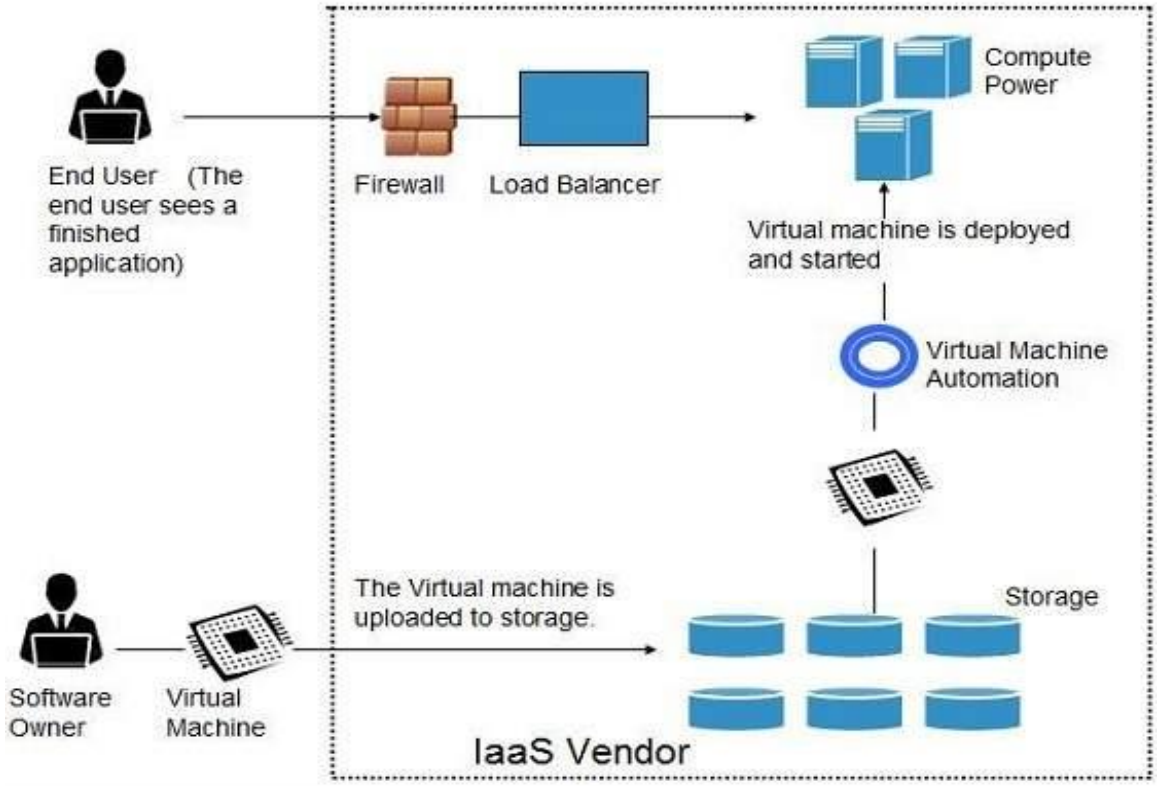
- Έξυπνη παρακολούθηση – διαχείριση: για να υπάρχει διαφάνεια και κλιμάκωση είναι απαραίτητη η έξυπνη διαχείριση και παρακολούθηση των εφαρμογών που διανέμονται.
- Ασφάλεια: το κέντρο αποθήκευσης δεδομένων πρέπει να διαθέτει ασφαλή αρχιτεκτονική. Ο κόμβος ελέγχου, ένα σημείο εισόδου στο κέντρο αποθήκευσης δεδομένων πρέπει επίσης να είναι ασφαλής.



Εικόνα 15: Χαρακτηριστικά υποδομών νέφους.

2.3 Μοντέλα Υπηρεσιών

Υποδομή σαν Υπηρεσία (Infrastructure as a Service – IaaS)



Εικόνα 16: Υποδομή σαν υπηρεσία.

Η υποδομή σαν υπηρεσία παρέχει πρόσβαση σε θεμελιώδεις πόρους όπως οι φυσικές μηχανές, οι εικονικές μηχανές, κλπ. Πέρα όμως από αυτά η υποδομή σαν υπηρεσία προσφέρει (Εικόνα 16):

- Αποθήκευση σε δίσκο εικονικής μηχανής
- Εικονικό τοπικό δίκτυο
- Εξισορροπιστές φόρτου
- IP Διευθύνσεις
- Πακέτα λογισμικού

Όλοι οι παραπάνω πόροι γίνονται διαθέσιμοι στους τελικούς χρήστες μέσω του εξυπηρετητή εικονικοποίησης. Επιπλέον, αυτοί οι πόροι προσπελαύνονται από τους πελάτες σαν αν είναι κάτοχοι οι ίδιοι.

Η υποδομή σαν υπηρεσία επιτρέπει στον πάροχο του νέφους να τοποθετήσει ελεύθερα την υποδομή πάνω από το Internet με αποδοτικό τρόπο από άποψη κόστους. Μερικά από τα πιο σημαντικά οφέλη της υποδομής σαν υπηρεσία είναι τα παρακάτω:

- Πλήρης έλεγχος των υπολογιστικών πόρων διαμέσου πρόσβασης διαχείρισης επάνω στις εικονικές μηχανές.
- Ευέλικτη και αποδοτική ενοικίαση υπολογιστικού υλικού.
- Φορητότητα, διαλειτουργικότητα. Για παράδειγμα, δικτυακές εφαρμογές όπως το e-mail, ή ο web server που κανονικά εκτελούνται πάνω από υλικού που κατέχει ο ίδιος ο χρήστης, μπορούν να εκτελούνται από τις εικονικές μηχανές μιας υποδομής σαν υπηρεσία.

Η Amazon είναι από τις πρωτοπόρες εταιρείες σε μοντέλα IaaS. Στα μέσα της δεκαετίας του 2000 παρουσίασε το Amazon Web Services (AWS), βασισμένο στο IaaS μοντέλο διανομής. Στο μοντέλο αυτό ο πάροχος των υπηρεσιών νέφους προσφέρει υποδομή που αποτελείται από εξυπηρετητές υπολογισμού και αποθήκευσης που διασυνδέονται μέσω ενός δικτύου υψηλών ταχυτήτων που υποστηρίζουν ένα σύνολο υπηρεσιών για να προσπελάσουν αυτούς τους πόρους. Ο κατασκευαστής εφαρμογών είναι υπεύθυνος για την εγκατάσταση εφαρμογών στην πλατφόρμα της επιλογής του και για τη διαχείριση των πόρων που παρέχονται από την Amazon [M2013].

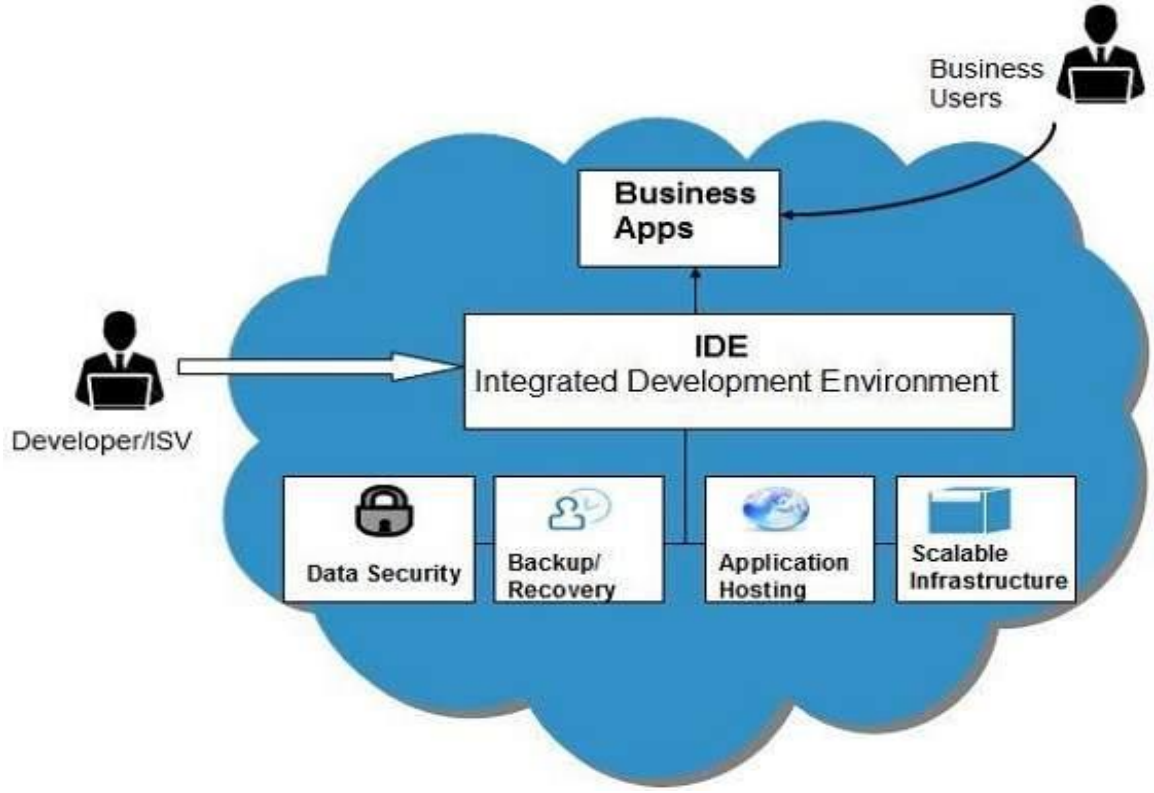
Η Elastic Compute Cloud (EC2) είναι μια υπηρεσία ιστού με απλή διεπαφή για την εκκίνηση στιγμιοτύπων εφαρμογής μέσα από αρκετά λειτουργικά συστήματα, όπως αρκετές Linux διανομές, Microsoft Windows Server 2003 και 2008, OpenSolaris, FreeBSD και NetBSD. Ένα στιγμιότυπο δημιουργείται είτε από μία προκαθορισμένη Amazon Machine Image (AMI) που είναι ψηφιακά υπογεγραμμένη και αποθηκευμένη στο Simple Storage System (S3) είτε από μία ορισμένη από το χρήστη εικόνα. Η εικόνα περιλαμβάνει το λειτουργικό σύστημα, το περιβάλλον εκτέλεσης, τις βιβλιοθήκες και την εφαρμογή που επιθυμεί ο χρήστης. Η υπηρεσία EC2 βασίζεται σε εικονικοποίηση τύπου Xen. Κάθε εικονική μηχανή ή στιγμιότυπο λειτουργεί σαν εικονικός ιδιωτικός εξυπηρετητής. Ένα στιγμιότυπο καθορίζει το μέγιστο πλήθος πόρων που είναι διαθέσιμοι για την εφαρμογή, τη διεπαφή για το στιγμιότυπο αυτό και το κόστος ανά ώρα. Περισσότερα όμως για αυτό το είδος εικονικοποίησης ο αναγνώστης μπορεί να διαβάσει σε επόμενη ενότητα.

Ένας χρήστης μπορεί να αλληλεπιδράσει με την EC2 χρησιμοποιώντας ένα σύνολο SOAP μηνυμάτων και να δει μια λίστα από διαθέσιμες AMI εικόνες, να εκκινήσει στιγμιότυπο μιας εικόνας, να τερματίσει μια εικόνα, να απεικονίσει τα εκτελούμενα στιγμιότυπα του χρήστη, κλπ. Ο χρήστης έχει δικαιώματα διαχειριστή σε κάθε στιγμιότυπο του ασφαλούς υπολογιστικού περιβάλλοντος της EC2.

Το CloudWatch είναι μια υποδομή παρακολούθησης που χρησιμοποιείται από τους κατασκευαστές εφαρμογών, τους χρήστες και τους διαχειριστές συστημάτων για να

συλλέγουν και να καταγράφουν μετρικές που είναι σημαντικές για τη βελτιστοποίηση της απόδοσης των εφαρμογών και την αύξηση της αποδοτικότητας της χρησιμοποίησης των πόρων.

Πλατφόρμα σαν Υπηρεσία (Platform as a Service)



Εικόνα 17: Πλατφόρμα σαν υπηρεσία.

Η πλατφόρμα σαν υπηρεσία προσφέρει το περιβάλλον εκτέλεσης για εφαρμογές. Επίσης, προσφέρει εργαλεία δημιουργίας και ανάπτυξης που απαιτούνται για την υλοποίηση εφαρμογών. Η πλατφόρμα σαν υπηρεσία έχει το χαρακτηριστικό της παροχής point-and-click εργαλείων που επιτρέπουν σε χρήστες που δεν γνωρίζουν πώς να προγραμματίζουν, να κατασκευάζουν web εφαρμογές. Το App Engine της Google και το Forces.com είναι παραδείγματα πλατφόρμας σαν υπηρεσία. Ο χρήστης που αναπτύσσει λογισμικό μπορεί να συνδεθεί σε αυτά τα websites και να χρησιμοποιήσει το built-in API που προσφέρεται για να δημιουργήσει web εφαρμογές. Το μειονέκτημα είναι ότι σε μια τέτοια περίπτωση ο προγραμματιστής εξαρτάται από τον πάροχο της PaaS.

Η Εικόνα 17 με ποιο τρόπο μία πλατφόρμα σαν υπηρεσία προσφέρει API και εργαλεία και πώς βοηθά τον τελικό χρήστη στην προσπέλαση επιχειρηματικών εφαρμογών.

Τα οφέλη από ένα τέτοιο μοντέλο είναι τα παρακάτω:

- Χαμηλότερη διαχειριστική επιβάρυνση, η οποία είναι αρμοδιότητα του παρόχου.
- Χαμηλότερο κόστος, καθώς ο χρήστης δεν χρειάζεται να αγοράσει ακριβό υλικό, εξυπηρετητές, ισχύ και χώρο αποθήκευσης δεδομένων.
- Κλιμακούμενες λύσεις, καθώς ο χρήστης μπορεί να αυξομειώνει τους πόρους που χρησιμοποιεί αυτόματα, αναλόγως της ζήτησης.
- Χρησιμοποίηση τρεχουσών εκδόσεων λογισμικού, το οποίο είναι αρμοδιότητα του παρόχου.

Όπως το λογισμικό σαν υπηρεσία, έτσι και η πλατφόρμα σαν υπηρεσία θέτει σημαντικούς περιορισμούς στους φυλλομετρητές των πελατών για τη διατήρηση αξιόπιστων και ασφαλών συνδέσεων με τα συστήματα του παρόχου. Έτσι υπάρχουν κάποια σημαντικά θέματα που σχετίζονται με την πλατφόρμα σαν υπηρεσία, όπως:

- Έλλειψη φορητότητας ανάμεσα σε νέφη PaaS. Παρόλο που χρησιμοποιούνται standard γλώσσες, οι υλοποιήσεις μπορεί να διαφέρουν. Για παράδειγμα, οι διεπαφές αρχείων, ουρών, ή πινάκων κατακερματισμού (hash tables) μπορεί να διαφέρουν μεταξύ των PaaS, κάνοντας δύσκολη τη μετακίνηση φόρου εργασίας από μία πλατφόρμα σε άλλη.
- Χρονοπρογραμματισμός επεξεργαστή με βάση τα γεγονότα: οι PaaS εφαρμογές είναι οδηγούμενες από τα γεγονότα, κάτι το οποίο θέτει στις εφαρμογές περιορισμούς ως προς τους πόρους. Για παράδειγμα, οι εφαρμογές πρέπει να απαντήσουν τις αιτήσεις σε ένα συγκεκριμένο χρονικό διάστημα.
- Ασφάλεια εφαρμογών. Από τη στιγμή που οι εφαρμογές βασίζονται στο δίκτυο, πρέπει οπωσδήποτε να χρησιμοποιούν κρυπτογραφία και να διαχειριστούν τη δημόσια έκθεση.

Λογισμικό σαν Υπηρεσία (Software as a Service – SaaS)

Το Λογισμικό σαν Υπηρεσία είναι ο κυρίαρχος τρόπος πλέον για τη διανομή λογισμικού. Εκτός του να είναι ένα μοντέλο διανομής εφαρμογών είναι και ένα επιχειρησιακό μοντέλο. Συγκεκριμένα, περιλαμβάνει ένα ευρύ φάσμα επιχειρήσεων, αγοράς και τεχνικές ευκαιρίες, καθώς και ζητήματα και προκλήσεις.

Η προσπάθεια της Google επικεντρώνεται στην περιοχή του Software as a Service (SaaS). Υπηρεσίες όπως Gmail, Google Drive, Google Calendar, Picasa και Google Groups παρέχονται δωρεάν για απλούς χρήστες και με κάποιο μικρό ποσό για οργανισμούς. Αυτές οι υπηρεσίες εκτελούνται στο νέφος και μπορούν να επικληθούν από ένα ευρύ φάσμα συσκευών, συμπεριλαμβανομένων κινητών συσκευών όπως iPhones, iPads, Black-Berrys, φορητούς υπολογιστές και tablets. Τα δεδομένα για αυτές τις συσκευές αποθηκεύονται σε κέντρα αποθήκευσης δεδομένων στο νέφος.

Η εξέλιξη του Λογισμικού σαν Υπηρεσία

Το Λογισμικό σαν Υπηρεσία έχει την ιδιότητα να είναι ένα μοντέλο διανομής εφαρμογών είναι και ένα επιχειρησιακό μοντέλο ταυτόχρονα. Επιτρέπει σε ένα πελάτη να χρησιμοποιήσει μια εφαρμογή με ένα τύπο πληρωμής pay-as-you-go και περιορίζει την ανάγκη για εγκατάσταση και εκτέλεση της εφαρμογής στη μεριά του υλικού του πελάτη. Οι πελάτες προσπελαύνουν την εφαρμογή μέσω ενός φυλλομετρητή ιστοσελίδων ή ενός thin client πάνω από το Internet. Το Λογισμικό σαν Υπηρεσία είναι πιο συχνά συνδρομητικό και η απαιτούμενη υποστήριξη, συντήρηση και αναβάθμιση αφήνεται στη μεριά του κατασκευαστή του λογισμικού σαν κομμάτι του λογισμικού. Οι δυνατότητες προσαρμογής (customization) της εφαρμογής, εάν υπάρχουν, παρέχονται σε όλους τους πελάτες με συνεπή τρόπο. Από τη σκοπιά του κατασκευαστή του λογισμικού το Λογισμικό σαν Υπηρεσία παρέχει μεγαλύτερη προστασία όσον αφορά τα πνευματικά του δικαιώματα, λειτουργικό έλεγχο του περιβάλλοντος όπου εκτελείται το λογισμικό και γενικά μια συνεχή ροή κέρδους από τις συνδρομές. Οι κατασκευαστές λογισμικού έχουν διάφορες δυνατότητες και οι εφαρμογές μπορεί να είναι ποικίλες, αλλά οι εφαρμογές τύπου Λογισμικό Σαν Υπηρεσία υποστηρίζουν πολλούς ξεχωριστούς πελάτες χρησιμοποιώντας ένα στιγμιότυπο αυτής της εφαρμογής.

Το μοντέλο εξέλιξης των τεσσάρων βημάτων

Η Saugatuck Technology όρισε τη θεωρία εξέλιξης των τριών βημάτων της αγοράς Λογισμικού Σαν Υπηρεσία. Σε μία πιο σύγχρονη αναφορά επέκτειναν το μοντέλο αυτό περιλαμβάνοντας και ένα τέταρτο βήμα. Η έννοια του μοντέλου των τεσσάρων βημάτων παρέχει μια χρήσιμη προσέγγιση για την κατανόηση του επιτακτικού και ώριμου SaaS μοντέλου και μπορεί να αποτελέσει ένα βοήθημα για τον κατασκευαστή λογισμικού ο οποίος επιθυμεί να μετακινήσει μια εφαρμογή βασισμένη σε ASP ή client – server μοντέλο σε SaaS.

Στο πρώτο βήμα οι κατασκευαστές SaaS παρέχουν πολύ απλές προσφορές που συνήθως περιλαμβάνουν stand alone εφαρμογές και ο στόχος είναι η μείωση του κόστους λόγω της ευκολίας της ανάπτυξης, συντήρησης και χρήσης. Σύγχρονες αναφορές δείχνουν ότι μόνο το 20% των προϋπολογισμών για Τεχνολογίες Πληροφορίας (IT) κατευθύνονται για το λογισμικό, ενώ τα υπόλοιπα χρήματα αφορούν ανθρώπους, υπηρεσίες και υλικό για να εκτελείται το λογισμικό αυτό. Από τη στιγμή που οι τελικοί χρήστες αυτό που πραγματικά θέλουν είναι το λογισμικό και όχι η επιβάρυνση που σχετίζεται με αυτό, υπάρχει μια αναντιστοιχία ανάμεσα στο κόστος και την αξία του προϊόντος που παρέχεται. Το πρώτο εξελικτικό βήμα αντιμετωπίζει ακριβώς αυτή την αναντιστοιχία.

Το δεύτερο εξελικτικό βήμα, το τέλος του οποίου είναι το σημείο όπου βρισκόμαστε τη στιγμή που γράφεται η παρούσα αναφορά, υπάρχει μια περισσότερο κυρίαρχη υιοθέτηση για τα SaaS και οι λύσεις είναι πιο εξελιγμένες. Καθώς οι SaaS παίζουν όλο και

μεγαλύτερο ρόλο στο συνολικό εταιρικό οικοσύστημα, η δυνατότητα διασύνδεσης όλων αυτών των λύσεων σαν ένα μέρος της συνολικής ροής δεδομένων γίνεται όλο και πιο σημαντική.

Στο τρίτο βήμα το Λογισμικό Σαν Υπηρεσία εξελίσσεται, ωριμάζει και κερδίζει ευρύτερη αποδοχή στο σημείο που γίνεται κεντρικό σημείο της συνολικής στρατηγικής λογισμικού μιας επιχείρησης. Σαν αποτέλεσμα γίνεται παράγοντας μετασχηματισμού μιας επιχείρησης. Η ολοκλήρωση SaaS υπηρεσιών και υπηρεσιών βάσης, όπως και ροή δεδομένων εντός της εταιρείας, είναι από τα σημαντικά χαρακτηριστικά αυτού του βήματος. Ένα πολύ καλό παράδειγμα είναι η ολοκλήρωση μιας εφαρμογής διαχείρισης ανθρώπινων πόρων όπως η Oracle PeopleSoft με μία SaaS CRM εφαρμογή όπως η Salesforce.com. Η εξάρτηση SaaS υπηρεσιών που ολοκληρώνονται σε μια ήδη υπάρχουσα δομή πηγών δεδομένων επιτρέπει το μετασχηματισμό της ίδιας της επιχείρησης, κάνοντας τις ήδη υπάρχουσες διεργασίες περισσότερο αποδοτικές και αποτελεσματικές από άποψη κόστους, και μερικές φορές δημιουργούν νέες διεργασίες που μέχρι εκείνη τη στιγμή ήταν αδύνατο να δημιουργηθούν.

Το τέταρτο βήμα αντιμετωπίζει την επιτακτικότητα του υπολογισμού νέφους και υποθέτει ότι η ισχυροποίηση αυτών των νέων περιβαλλόντων είναι η φυσική κατάληξη του SaaS, η φιλοσοφία της ωφελιμότητας της Τεχνολογίας της Πληροφορίας, και η επιχειρηματική διεργασία εξωτερικής ανάθεσης (outsourcing) και μετασχηματισμού. Ο υπολογισμός σε νέφος προσφέρει υποδομή κατ' απαίτηση σε συνδυασμό με λογισμικό κατ' απαίτηση και έτσι παρέχει περισσότερο ευέλικτες στην ανάπτυξη και τη διαχείριση λύσεις, εξυπηρετώντας έτσι την επιτάχυνση του μετασχηματισμού της εταιρείας που είδαμε στο τρίτο βήμα.

Ταυτότητα σαν Υπηρεσία (Identity as a Service – IDaaS)

Οι υπάλληλοι μιας εταιρίας χρειάζεται να εισαχθούν σε ένα σύστημα για να εκτελέσουν διάφορες εργασίες. Αυτά τα συστήματα μπορεί να βασίζονται σε ένα τοπικό εξυπηρετητή ή σε νέφος. Έτσι μπορεί να υπάρξουν τα παρακάτω προβλήματα:

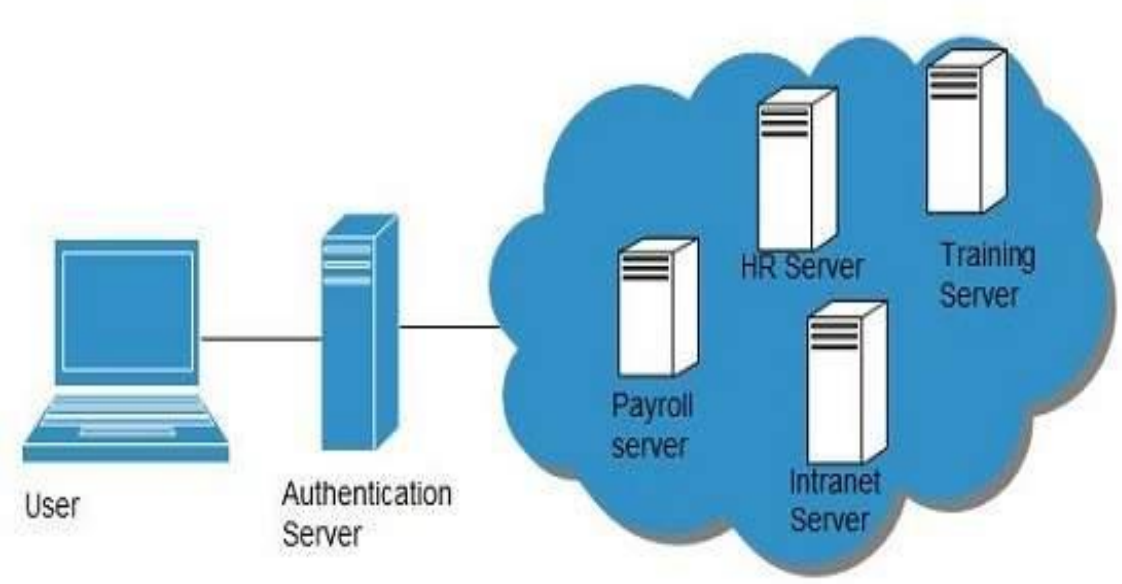
1. Ένας υπάλληλος να χρειάζεται να θυμάται διαφορετικούς συνδυασμούς usernames και passwords προκειμένου να έχει πρόσβαση σε διάφορους εξυπηρετητές.
2. Όταν ένας υπάλληλος εγκαταλείπει την εταιρία, είναι απαραίτητο να επιβεβαιωθεί ότι όλοι λογαριασμοί του έχουν απενεργοποιηθεί.

Η ταυτότητα σαν υπηρεσία προσφέρει διαχείριση πληροφορίας ταυτότητας σαν μια ψηφιακή οντότητα. Η ταυτότητα αυτή μπορεί να χρησιμοποιηθεί κατά τη διάρκεια ηλεκτρονικών συναλλαγών.

Για τη λύση του προβλήματος της χρήσης διαφορετικών συνδυασμών usernames και passwords για διαφορετικούς εξυπηρετητές, οι εταιρίες πλέον υιοθετούν ένα λογισμικό

Single Sign-On, όπου ο χρήστης μπορεί να εισαχθεί μόνο μία φορά σε ένα σύστημα και διαχειρίζεται την πρόσβαση σε άλλα συστήματα.

Το σύστημα αυτό διαθέτει ένα μοναδικό εξυπηρετητή πιστοποίησης που διαχειρίζεται την πολλαπλή πρόσβαση σε άλλα συστήματα, όπως φαίνεται στην Εικόνα 18.

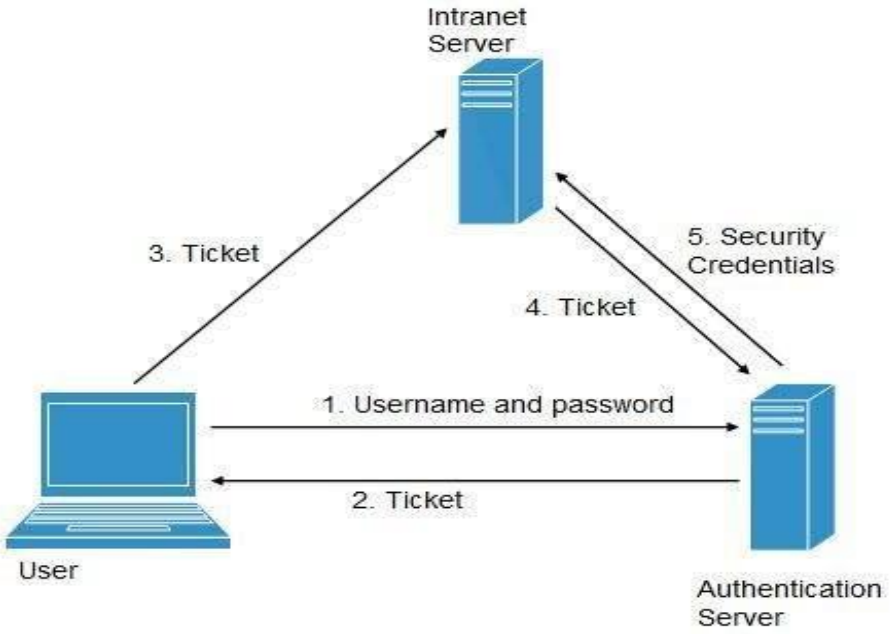


Εικόνα 18: Single Sign-On.

Η συνήθης υλοποίηση για Single Sign-On παρουσιάζεται στην Εικόνα 19. Σύμφωνα με το σχήμα αυτό:

1. ο χρήστης εισάγεται στον εξυπηρετητή πιστοποίησης χρησιμοποιώντας ένα username και ένα password.
2. Ο εξυπηρετητής πιστοποίησης του επιστρέφει ένα εισιτήριο.
3. Ο χρήστης στέλνει το εισιτήριο στον intranet εξυπηρετητή.
4. Ο intranet εξυπηρετητής στέλνει το εισιτήριο στον εξυπηρετητή πιστοποίησης.
5. Ο εξυπηρετητής πιστοποίησης στέλνει τα διαπιστευτήρια του χρήστη για αυτόν τον εξυπηρετητή πίσω στον intranet εξυπηρετητή.

Η τεχνική Federated Identity Management (FIDM) περιγράφει τις τεχνολογίες και τα πρωτόκολλα που επιτρέπουν σε ένα χρήστη να τοποθετήσει σε ένα πακέτο τα διαπιστευτήρια ασφάλειας. Για το λόγο αυτό χρησιμοποιεί τη γλώσσα Security Markup Language (SAML).



Εικόνα 19: Υλοποίηση Single Sign-On.

Δίκτυο σαν Υπηρεσία (Network as a Service)

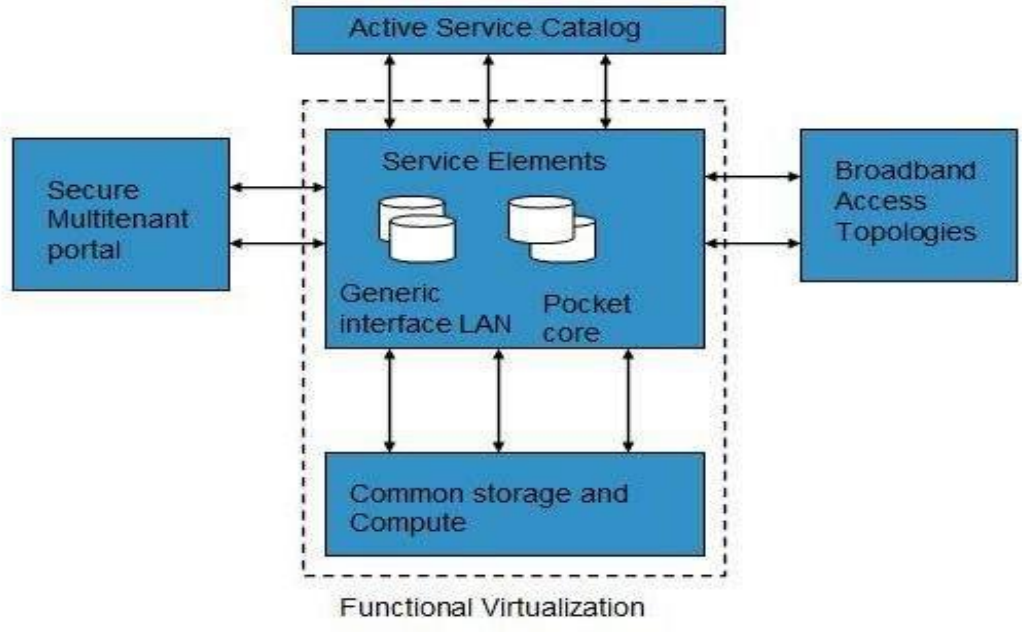
Το Δίκτυο σαν Υπηρεσία επιτρέπει την πρόσβαση στη δικτυακή υποδομή με τρόπο άμεσο και ασφαλή. Το Δίκτυο σαν Υπηρεσία κάνει δυνατή την ανάπτυξη πρωτοκόλλων δρομολόγησης κατά περίπτωση σύμφωνα με τις ανάγκες του πελάτη. Χρησιμοποιεί εικονικοποιημένη δικτυακή υποδομή για την παροχή δικτυακών υπηρεσιών στον πελάτη. Είναι αρμοδιότητα του παρόχου η συντήρηση και η διαχείριση των δικτυακών πόρων. Ο πελάτης έχει τη δυνατότητα να πληρώνει κάθε φορά μόνο για τη χωρητικότητα δικτύου που χρησιμοποιεί. Υπάρχει επίσης δυνατότητα ο πελάτης να μειώνει τη χρησιμοποιούμενη χωρητικότητα οποιαδήποτε στιγμή.

Το Δίκτυο σαν Υπηρεσία για κινητά δίκτυα προσφέρει ευέλικτο και αποδοτικό έλεγχο των δικτυακών συσκευών. Η Εικόνα 20 παρουσιάζει τα συστατικά στοιχεία μιας NaaS για κινητά δίκτυα.

Το Δίκτυο σαν Υπηρεσία προσφέρει διάφορα οφέλη, όπως φαίνονται παρακάτω:

- Ανεξαρτησία μεταξύ πελατών.
- Πληρωμή για δίκτυο υψηλής χωρητικότητας κατ’ απαίτηση.
- Ελαστικότητα: υπάρχει αξιοπιστία εξυπηρέτησης, η οποία μπορεί να εφαρμοστεί και για κρίσιμες εφαρμογές.

- Προστασία για εφαρμογές με ευαίσθητα δεδομένα.
- Ευκολία εγκατάστασης νέων υπηρεσιών.
- Υπάρχει πληθώρα υποστηρικτών μοντέλων για τη μείωση του λειτουργικού κόστους.
- Απομόνωση «κυκλοφορίας» για κάθε πελάτη



Εικόνα 20: Δίκτυο σαν Υπηρεσία για κινητά δίκτυα.

2.4 Διαχείριση

Είναι αρμοδιότητα του παρόχου η διαχείριση των πόρων και η απόδοσή τους. Η διαχείριση των πόρων περιλαμβάνει αρκετά θέματα όπως εξισορρόπηση φόρτου, απόδοση, αποθήκευση, αντίγραφα ασφαλείας, χωρητικότητα, ανάπτυξη, κλπ.

Ο πάροχος του νέφους εκτελεί διάφορες εργασίες για την επιβεβαίωση της αποδοτικής χρήσης των πόρων. Εδώ συζητάμε για μερικές από αυτές.

- Σύστημα παρακολούθησης αντιγράφων. Είναι απαραίτητη η παρακολούθηση στο χρόνο των αντιγράφων ασφαλείας για εξασφάλιση της αποθήκευσης τυχαία επιλεγμένων αρχείων διαφορετικών χρηστών. Αντίγραφα ασφαλείας μπορούν να εκτελεστούν με ένα από τους ακόλουθους τρόπους:

1. Αντίγραφα αρχείων από την εταιρία, από τους δικούς της υπολογιστές προς τους δίσκους του νέφους.

2. Αντίγραφα αρχείων που παίρνονται από τον πάροχο.

Είναι απαραίτητο ο χρήστης να γνωρίζει, εάν ο πάροχος έχει κρυπτογραφήσει τα δεδομένα, ποιος έχει πρόσβαση σε αυτά και εάν τα αντίγραφα ασφαλείας παίρνονται σε διαφορετικές τοποθεσίες, λεπτομέρειες για τις θέσεις αυτές.

- Ροή δεδομένων στο σύστημα. Οι διαχειριστές είναι απαραίτητο να κατασκευάσουν ένα διάγραμμα που να περιγράφει με λεπτομέρειες τη ροή διεργασιών. Αυτή η ροή διεργασιών περιγράφει τη μετακίνηση δεδομένων ενός οργανισμού εντός του νέφους.
- Εξάρτηση από κατασκευαστή και λύσεις. Οι διαχειριστές πρέπει να γνωρίζουν τη διαδικασία εξόδου από υπηρεσίες ενός συγκεκριμένου παρόχου νέφους. Πρέπει να οριστούν οι διαδικασίες που να επιτρέπουν στους διαχειριστές του νέφους να εξάγουν τα δεδομένα ενός οργανισμού από το σύστημά τους στο σύστημα ενός άλλου παρόχου.
- Γνώση των διαδικασιών ασφαλείας του παρόχου. Οι διαχειριστές πρέπει να γνωρίζουν τα πλάνα ασφάλειας του παρόχου για τις εξής υπηρεσίες: χρήση από πολλαπλούς πελάτες, διεργασίες ηλεκτρονικού εμπορίου, παρακολούθηση υπαλλήλων, πολιτική κρυπτογράφησης.
- Παρακολούθηση πλάνου χωρητικότητας και δυνατότητες κλιμάκωσης. Οι διαχειριστές πρέπει να γνωρίζουν το πλάνο χωρητικότητας ώστε να επιβεβαιώσουν ότι ο πάροχος θα ικανοποιήσει τις μελλοντικές απαιτήσεις χωρητικότητας για αυτή την επιχείρηση. Πρέπει ακόμη να εξασφαλίσουν ότι οι υπηρεσίες μπορούν να κλιμακωθούν είτε προς τα πάνω είτε προς τα κάτω σύμφωνα με τις ανάγκες του πελάτη.
- Για την ταυτοποίηση σφαλμάτων στο σύστημα, οι διαχειριστές πρέπει να παρακολουθούν τις εγγραφές στα απαραίτητα log files, σε τακτική χρονική βάση.
- Έλεγχος λύσεων και επιβεβαίωση. Όταν ο πάροχος του νέφους προσφέρει μία λύση, είναι απαραίτητο να την έχει πρώτα ελέγξει ώστε να εξασφαλίσει ότι δίνει το σωστό αποτέλεσμα και δεν παράγει σφάλματα. Αυτό είναι απαραίτητο για το σύστημα ώστε αυτό να είναι εύρωστο και αξιόπιστο.

2.5 Αποθήκευση Δεδομένων

Η αποθήκευση σε νέφος είναι μια υπηρεσία που επιτρέπει την αποθήκευση δεδομένων που ζητούν οι διάφορες εφαρμογές σε ένα σύστημα αποθήκευσης εκτός του site του πελάτη, που το διαχειρίζεται μια τρίτη οντότητα (third party) και είναι προσβάσιμο μέσω ενός API υπηρεσιών ιστού. Αυτό γίνεται κατορθωτό λόγω του δικτυο-κεντρικού μοντέλου αποθήκευσης περιεχομένου που ακολουθείται στα συστήματα νέφους, σύμφωνα με το οποίο οι χρήστες μπορούν να προσπελάσουν δεδομένα αποθηκευμένα στο νέφος από οποιαδήποτε συσκευή συνδεδεμένη στο internet. Κινητές συσκευές με περιορισμένες δυνατότητες ισχύος και τοπικής αποθήκευσης μπορούν να αποθηκεύουν

αρχεία ήχου και video στο νέφος. Έτσι τα νέφη παρέχουν ένα ιδανικό περιβάλλον για διανομή πολυμεσικού περιεχομένου.

Μια μεγάλη ποικιλία εισόδων τροφοδοτούν με συνεχόμενη ροή δεδομένων τις εφαρμογές του νέφους. Ένας συνεχώς αυξανόμενος αριθμός από υπηρεσίες βασισμένες στο νέφος συλλέγουν λεπτομερή δεδομένα για τις υπηρεσίες τους και πληροφορίες για τους χρήστες αυτών των υπηρεσιών. Έπειτα οι πάροχοι των υπηρεσιών χρησιμοποιούν τα νέφη για να αναλύσουν τα δεδομένα [M2013].

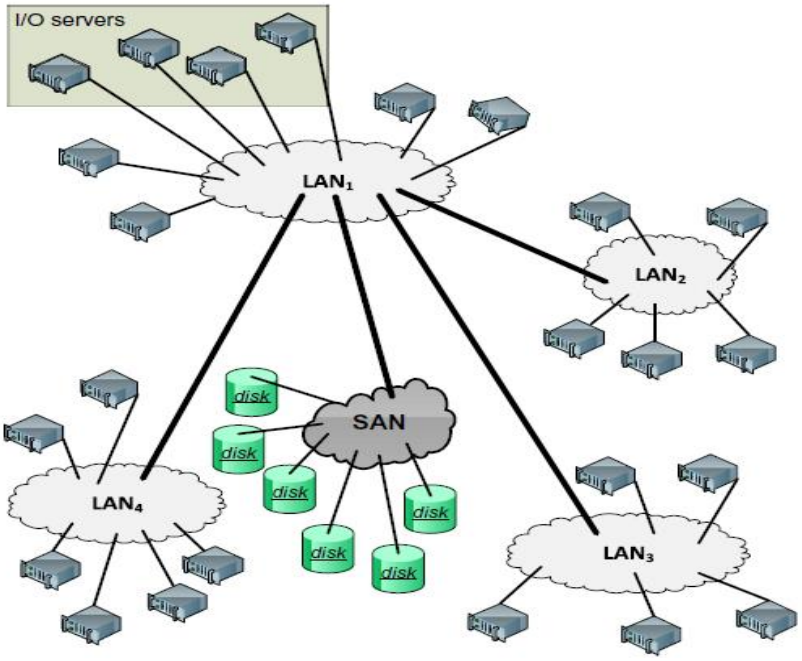
Αποθήκευση και επεξεργασία στο νέφος είναι έννοιες αλληλένδετες. Μάλιστα, είναι απαραίτητη χρήση προχωρημένων στρατηγικών για τη μείωση του χρόνου πρόσβασης και την υποστήριξη της πρόσβασης σε πραγματικό χρόνο σε πολυμέσα για την ικανοποίηση των απαιτήσεων διανομής περιεχομένου. Από την άλλη, οι περισσότερες εφαρμογές νέφους επεξεργάζονται τεράστια μεγέθη δεδομένων. Μάλιστα, οι στρατηγικές για την αποδοτική δημιουργία αντιγράφων δεδομένων και διαχείρισης αποθηκευτικού χώρου είναι πολύ κρίσιμες για τους υπολογισμούς στο νέφος.

Ο νέος όρος, «μεγάλα δεδομένα (big data)», αντανακλά το γεγονός ότι πολλές εφαρμογές χρησιμοποιούν σύνολα δεδομένων που είναι τόσο μεγάλα που δεν μπορούν να τα αποθηκεύσουν ούτε να τα επεξεργαστούν οι τοπικές συσκευές με τους πόρους που διαθέτουν. Εφαρμογές διαφόρων επιστημονικών πεδίων, όπως γονιδιωματική, ανατομική βιολογία, φυσική υψηλής ενέργειας, αστρονομία, μετεωρολογία, μελέτη περιβάλλοντος, εκτελούν περίπλοκες αναλύσεις συνόλων δεδομένων, πολύ συχνά της τάξης των terabytes.

Οι συσκευές αποθήκευσης μπορούν ευρέως να χωριστούν σε δύο κατηγορίες: συσκευές αποθήκευσης τμημάτων (block storage devices) και συσκευές αποθήκευσης αρχείων (file storage devices). Οι πρώτες προσφέρουν ακατέργαστη αποθήκευση στους πελάτες. Αυτές οι ακατέργαστες αποθηκεύσεις διαμερίζονται σε τόμους. Οι δεύτερες προσφέρουν αποθήκευση σε μορφή αρχείων επιτρέποντας στον καθένα να διατηρεί το δικό του σύστημα αρχείων.

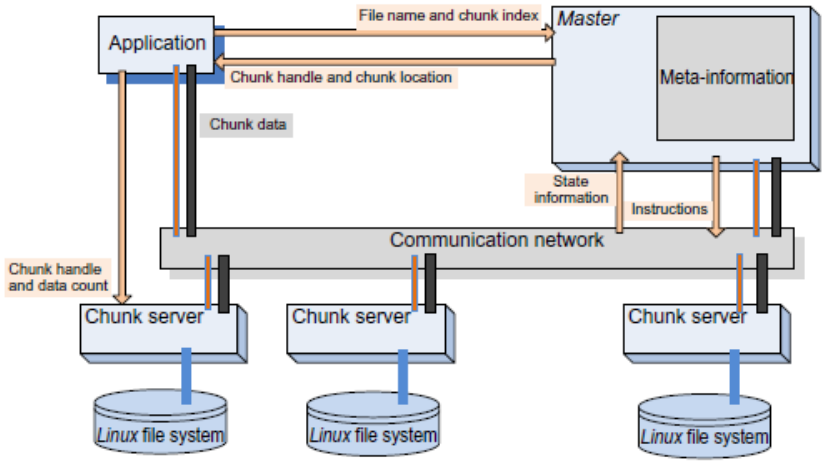
Η πρόοδος στην δικτυακή τεχνολογία επιτρέπει τον διαχωρισμό των συστημάτων αποθήκευσης από τους εξυπηρετητές υπολογισμών. Τα δύο αυτά μέρη μπορούν να διασυνδεθούν μέσω ενός Δικτύου Περιοχής Αποθήκευσης (Storage Area Network – SAN). Τα SAN προσφέρουν πρόσθετη ευελιξία και επιτρέπουν στους εξυπηρετητές του νέφους να αντιμετωπίζουν μη άτακτες αλλαγές στη διαμόρφωση της αποθήκευσης. Επιπλέον, η αποθήκευση σε SAN μπορεί να ενοποιηθεί και να εκχωρηθεί με βάση τις ανάγκες των εξυπηρετητών. Η ενοποίηση αυτή απαιτεί πρόσθετη υποστήριξη σε λογισμικό και υλικό και είναι ένα ακόμη πλεονέκτημα του κεντρικοποιημένου συστήματος αποθήκευσης. Μια υλοποίηση με SAN ενός συστήματος αρχείων μπορεί να είναι ακριβή, αφού κάθε κόμβος πρέπει να διαθέτει προσαρμογέα οπτικού καναλιού για συνδεθεί στο δίκτυο [M2013].

Τα παράλληλα συστήματα αρχείων είναι επεκτάσιμα, είναι ικανά στην κατανομή αρχείων κατά μήκος ενός μεγάλου αριθμού κόμβων και παρέχουν καθολικό χώρο ονοματολογίας. Στα παράλληλα συστήματα δεδομένων, αρκετοί κόμβοι Ε/Ε εξυπηρετούν δεδομένα για όλους τους υπολογιστικούς κόμβους και περιλαμβάνουν ένα εξυπηρετητή μεταδεδομένων που περιέχει πληροφορίες για τα δεδομένα που αποθηκεύονται στους κόμβους Ε/Ε. Η παράλληλη Ε/Ε απαιτεί την εκτέλεση πολλαπλών λειτουργιών Ε/Ε ταυτόχρονα. Η υποστήριξη παράλληλης Ε/Ε είναι βασική για την απόδοση πολλών εφαρμογών. Το δίκτυο διασύνδεσης ενός παράλληλου συστήματος αρχείων μπορεί να είναι ένα SAN (Εικόνα 21).



Εικόνα 21: Η αρχιτεκτονική ενός γενικού παράλληλου συστήματος αρχείων.

Ένα άλλο παράδειγμα κατανεμημένου συστήματος αρχείων είναι το Google File System (GFS), του οποίου η αρχιτεκτονική παρουσιάζεται στην (Εικόνα 22). Ο κόμβος master περιέχει πληροφορίες για όλα τα συστατικά μέρη του συστήματος, ουσιαστικά ελέγχει ένα αριθμό από chunk servers. Ένας chunk server διαθέτει linux λειτουργικό σύστημα και χρησιμοποιεί μεταδεδομένα που παίρνει από το master για να επικοινωνήσει απ' ευθείας με την εφαρμογή. Η ροή των δεδομένων εφαρμογής ξεχωρίζει από τη ροή δεδομένων ελέγχου. Στην Εικόνα 22τα μονοπάτια δεδομένων και ελέγχου ξεχωρίζουν. Τα μονοπάτια δεδομένων είναι αυτά με τις παχιές γραμμές, ενώ αυτά με τις λεπτές γραμμές είναι τα μονοπάτια ελέγχου [M2013].



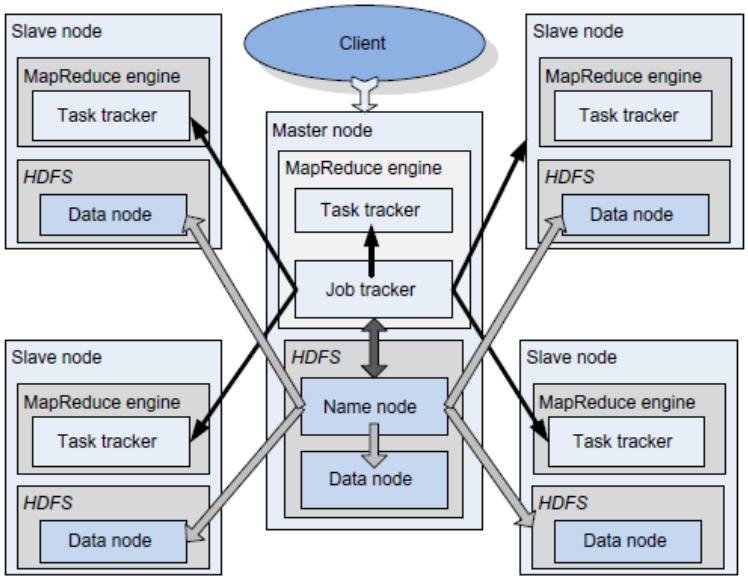
Εικόνα 22: Η αρχιτεκτονική του Google File System (GFS).

Μια μεγάλη γκάμα εφαρμογών που διαθέτουν ευαίσθητα δεδομένα όπως εφαρμογές για ανάλυση αγορών, επεξεργασία εικόνας, εκμάθηση μηχανών και web crawling χρησιμοποιούν το Apache Hadoop, ένα ανοικτού κώδικα βασισμένο σε Java σύστημα λογισμικού. Το Hadoop υποστηρίζει κατανεμημένες εφαρμογές που χειρίζονται εξαιρετικά μεγάλους όγκους δεδομένων. Η μακρά λίστα των χρηστών του Hadoop περιλαμβάνει πολύ μεγάλες εταιρείες όπως: Apple, IBM, HP, Microsoft, Yahoo! και Amazon, ειδησιογραφικές εταιρείες όπως: New York Times και Fox, κοινωνικά δίκτυα όπως: Twitter, Facebook και LinkedIn [M2013].

Το Hadoop σύστημα έχει δύο συστατικά, μια μηχανή MapReduce και μια βάση δεδομένων (Εικόνα 23). Η βάση δεδομένων θα μπορούσε να είναι το Hadoop File System (HDFS) ή το Amazon S3, ή το CloudStore (μια υλοποίηση του Google File System). Το HDFS είναι ένα κατανεμημένο σύστημα αρχείων γραμμένο σε Java, είναι φορητό, αλλά δεν μπορεί να προσαρτηθεί απ' ευθείας σε ένα υπάρχον λειτουργικό σύστημα. Το HDFS δεν είναι πλήρως συμβατό με το POSIX, αλλά είναι υψηλής απόδοσης.

Στην Εικόνα 23 παρουσιάζεται η αρχιτεκτονική ενός Hadoop cluster που χρησιμοποιεί το HDFS. Η ομάδα (cluster) περιλαμβάνει τέσσερις slave κόμβους και ένα master κόμβο. Κάθε κόμβος εκτελεί μια MapReduce μηχανή. Η μηχανή στον master κόμβο περιλαμβάνει ένα job tracker και ένα task tracker, ενώ η μηχανή ενός slave κόμβου έχει μόνο ένα task tracker. Ο job tracker λαμβάνει μια εργασία τύπου MapReduce από πελάτη και δρομολογεί τη δουλειά στους task trackers που εκτελούνται στους slave κόμβους. Για τη βελτίωση της απόδοσης, ο job tracker προσπαθεί να δρομολογήσει τα tasks προς τους διαθέσιμους slaves που είναι πιο κοντά στην περιοχή που βρίσκονται αποθηκευμένα τα δεδομένα. Ο task server επιβλέπει την εκτέλεση της εργασίας που εκχωρήθηκε στον

κόμβο. Το HDFS δημιουργεί αντίγραφα δεδομένων σε πολλούς κόμβους. Η προεκχωρημένη τιμή είναι για τρία αντίγραφα. Η υπηρεσία ονομάτων (name node) που εκτελείται στον master κόμβο διαχειρίζεται την κατανομή των δεδομένων και την αντιγραφή των δεδομένων και επικοινωνεί με τις υπηρεσίες δεδομένων (data node) που εκτελούνται σε όλους τους κόμβους της ομάδας, ενώ επίσης διαμοιράζεται με τον job tracker πληροφορίες για την τοποθέτηση των δεδομένων για την ελαχιστοποίηση της επικοινωνίας μεταξύ των κόμβων στους οποίους είναι τοποθετημένα τα δεδομένα και αυτούς που τα χρειάζονται. Τέλος, το HDFS μπορεί να χρησιμοποιηθεί και για εφαρμογές που δεν είναι βασισμένες στο MapReduce μοντέλο [M2013].



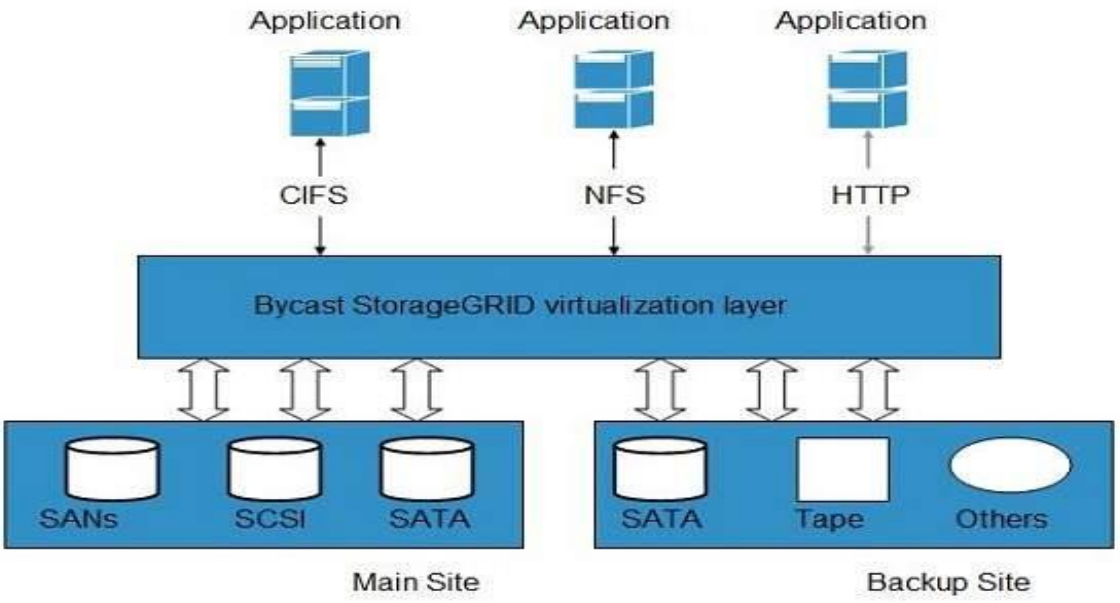
Εικόνα 23: Η αρχιτεκτονική του Hadoop.

Η αποθήκευση σε νέφος μπορεί να γενικά να κατηγοριοποιηθεί σε δύο ομάδες: μη διαχειρίσιμη αποθήκευση και διαχειρίσιμη αποθήκευση. Μη διαχείρισιμη είναι η αποθήκευση που έχει προδιαμορφωθεί για τον πελάτη. Δηλαδή, να εκτελέσει φορμάρισμα, ούτε να εγκαταστήσει το δικό του σύστημα αρχείων, ούτε να ρυθμίσει τις ιδιότητες των οδηγών δίσκων. Σε αντίθεση, η διαχειρίσιμη αποθήκευση επιτρέπει στον πελάτη να κάνει διαμέριση και φορμάρισμα.

Το σύστημα αποθήκευσης του νέφους προσφέρει πολλαπλά αντίγραφα δεδομένων σε πολλαπλούς εξυπηρετητές, σε πολλαπλές τοποθεσίες. Εάν ένα σύστημα καταρρεύσει, τότε απαιτείται απλά να αλλάξει ο δείκτης προς την τοποθεσία όπου βρίσκεται αποθηκευμένο το αντικείμενο.

Για τη συνένωση διαφορετικών στοιχείων αποθήκευσης, ο πάροχος του νέφους χρησιμοποιεί ένα λογισμικό εικονικοποιημένης αποθήκευσης που λέγεται

StorageGRID. Αυτό δημιουργεί ένα επίπεδο εικονικοποίησης που συνενώνει αποθήκευση από διαφορετικές συσκευές αποθήκευσης σε ένα ενιαίο σύστημα διαχείρισης. Μπορεί επίσης να διαχειριστεί δεδομένα από τα CIFS και NFS συστήματα αρχείων πάνω από το internet (Εικόνα 24).



Εικόνα 24: Σύστημα αποθήκευσης νέφους.

2.6 Εικονικοποίηση

Οι εικονικές μηχανές δημιουργούν ένα περιβάλλον που είναι ένα ξεχωριστό επίπεδο αφαίρεσης από το βασικό υλικό. Η μηχανή πάνω στην οποία δημιουργείται η εικονική μηχανή λέγεται host machine και η εικονική μηχανή λέγεται guest machine. Η εικονική μηχανή διαχειρίζεται από ένα λογισμικό ή firmware που λέγεται hypervisor και το οποίο διαμερίζει με ασφάλεια τους πόρους ενός υπολογιστικού συστήματος σε μία ή περισσότερες εικονικές μηχανές. Ένα φιλοξενούμενο λειτουργικό σύστημα (guest operating system) είναι ένα λειτουργικό σύστημα το οποίο εκτελείται υπό τον έλεγχο του hypervisor, αντί απ' ευθείας πάνω από το υλικό. Ο hypervisor εκτελείται σε τρόπο λειτουργίας συστήματος, ενώ το φιλοξενούμενο λειτουργικό σύστημα εκτελείται σε τρόπο λειτουργίας χρήστη. Κάποιες φορές το υλικό υποστηρίζει ένα τρίτο τρόπο λειτουργίας για την εκτέλεση του φιλοξενούμενου λειτουργικού συστήματος [M2013].

Οι hypervisors επιτρέπουν αρκετά λειτουργικά συστήματα να εκτελούνται ταυτόχρονα σε μία πλατφόρμα υλικού. Την ίδια χρονική στιγμή εξασφαλίζουν την απομόνωση μεταξύ αυτών των συστημάτων αυξάνοντας την ασφάλεια. Ο hypervisor ελέγχει τον τρόπο με τον οποίο το φιλοξενούμενο λειτουργικό σύστημα διαχειρίζεται τους πόρους

του υλικού. Τα γεγονότα που συμβαίνουν σε μία εικονική μηχανή δεν επηρεάζουν καμιά άλλη εικονική μηχανή που εκτελείται υπό τον ίδιο hypervisor. Την ίδια ώρα επιτρέπει:

- Πολλαπλές υπηρεσίες να διαμοιράζονται την ίδια πλατφόρμα.
- Τη μετακίνηση εξυπηρετητή από μία πλατφόρμα σε μία άλλη, γνωστή σαν μετακίνηση σε πραγματικό χρόνο.
- Τροποποίηση του συστήματος ενώ διατηρεί προς τα πίσω συμβατότητα με το αυθεντικό σύστημα.

Όταν ένα φιλοξενούμενο λειτουργικό σύστημα προσπαθεί να εκτελέσει μια εντολή που απαιτεί προνόμια, ο hypervisor δημιουργεί μια trap για τη λειτουργία αυτή και εφαρμόζει τους κανόνες ορθότητες και ασφάλειας της λειτουργίας. Ο hypervisor εγγυάται την απομόνωση της ατομικής εικονικής μηχανής και εξασφαλίζει την ασφάλεια και τη συμπερίληψη της λειτουργίας, που είναι μέγιστης σημασίας στον υπολογισμό νέφους. Την ίδια στιγμή ο hypervisor παρακολουθεί την απόδοση του συστήματος και αναλαμβάνει μια διορθωτική δράση προκειμένου να αποφευχθεί πτώση της απόδοσης. Για παράδειγμα, ο hypervisor μπορεί να θέσει εκτός συστήματος μια εικονική μηχανή (αντιγράφει όλες τις σελίδες της εικονικής μηχανής από την πραγματική μνήμη στο δίσκο και κάνει τα πλαίσια της πραγματικής μνήμης διαθέσιμα για σελιδοποίηση από άλλες εικονικές μηχανές) για την αποφυγή του λυγισμού.

Μια εικονική μηχανή εικονικοποιεί την μονάδα επεξεργασίας και τη μνήμη. Για παράδειγμα, η εικονική μηχανή δημιουργεί μια trap για ένα σήμα διακοπής και το δρομολογεί στο ανεξάρτητο φιλοξενούμενο λειτουργικό σύστημα. Εάν το φιλοξενούμενο λειτουργικό σύστημα έχει απενεργοποιήσει τα σήματα διακοπής, η εικονική μηχανή αποθηκεύει προσωρινά κάθε σήμα διακοπής μέχρι το φιλοξενούμενο λειτουργικό σύστημα να τα ενεργοποιήσει. Ο hypervisor διατηρεί ένα κρυφό πίνακα σελίδων (shadow page table) για κάθε φιλοξενούμενο λειτουργικό σύστημα και δημιουργεί αντίγραφο για κάθε τροποποίηση από το φιλοξενούμενο λειτουργικό σύστημα στο δικό του κρυφό πίνακα σελίδων. Αυτός ο κρυφός πίνακας σελίδων δείχνει στα πραγματικά πλαίσια των σελίδων και χρησιμοποιείται από το συστατικό του υλικού που καλείται Μονάδα Διαχείρισης Μνήμης (Memory Management Unit – MMU) για δυναμική μετάφραση διευθύνσεων.

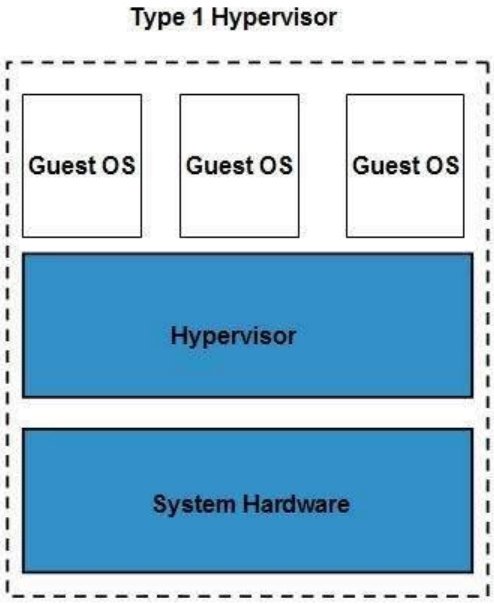
Η εικονικοποίηση της μνήμης έχει σημαντική επίδραση στην απόδοση. Οι hypervisors χρησιμοποιούν μια πλειάδα από τεχνικές βελτιστοποίησης, για παράδειγμα, το VMware αποφεύγει τον διπλασιασμό των σελίδων μεταξύ διαφορετικών εικονικών μηχανών, διατηρεί μόνο ένα αντίγραφο μιας διαμοιρασμένης σελίδας και χρησιμοποιεί την πολιτική αντιγραφής κατά την εγγραφή (copy on write), ενώ το Xen θέτει πλήρη απομόνωση μιας εικονικής μηχανής και δεν επιτρέπει διαμοίραση σελίδων. Οι hypervisors ελέγχουν τη διαχείριση εικονικής μνήμης και αποφασίζουν ποιες σελίδες θα δρομολογήσουν εκτός συστήματος (swap out).

Υπάρχουν δύο τύποι εικονικών μηχανών: διεργασίας και συστήματος. Μια εικονική μηχανή διεργασίας είναι μια εικονική πλατφόρμα που δημιουργείται για μια ξεχωριστή διεργασία και καταστρέφεται μόλις η διεργασία τερματίζει. Εικονικά όλα τα λειτουργικά συστήματα παρέχουν μία εικονική μηχανή διεργασίας για καθεμία από τις εφαρμογές που εκτελούνται, αλλά οι πιο ενδιαφέρουσες εικονικές μηχανές διεργασίας είναι που υποστηρίζουν δυαδική μετάφραση για ένα διαφορετικό σύνολο εντολών ενός επεξεργαστή. Μια εικονική μηχανή συστήματος υποστηρίζει ένα λειτουργικό σύστημα με πολλές διεργασίες χρηστών. Όταν η εικονική μηχανή εκτελείται υπό την επίβλεψη ενός κανονικού λειτουργικού συστήματος και παρέχει φιλοξενία ανεξαρτήτως πλατφόρμας για μια απλή εφαρμογή, έχουμε μια εικονική μηχανή εφαρμογής (π.χ. η Java Virtual Machine – JVM).

Μια εικονική μηχανή συστήματος παρέχει ένα πλήρες σύστημα, όπου κάθε εικονική μηχανή εκτελεί το δικό της λειτουργικό σύστημα, το οποίο με τη σειρά του μπορεί να εκτελεί πολλές εφαρμογές. Συστήματα όπως Linux Vserver, OpenVZ, FreeBSD Jails και Solaris Zones υλοποιούν τεχνολογίες εικονικοποίησης επιπέδου λειτουργικού συστήματος. Η εικονικοποίηση επιπέδου λειτουργικού συστήματος επιτρέπει σε ένα φυσικό εξυπηρετητή να εκτελεί πολλά απομονωμένα στιγμιότυπα λειτουργικού συστήματος, τα οποία υπόκεινται σε αρκετούς περιορισμούς. Τα στιγμιότυπα είναι γνωστά σαν containers, εικονικοί ιδιωτικοί εξυπηρετητές (virtual private servers) ή εικονικά περιβάλλοντα (virtual environments). Τα συστήματα αυτά υπερέχουν σε απόδοση έναντι των συστημάτων που βασίζονται σε ένα hypervisor όπως το Xen ή το VMware – το OpenVZ παρουσιάζει μόνο 1% ως 3% έλλειμμα απόδοσης σε σχέση με το stand-alone Linux server.

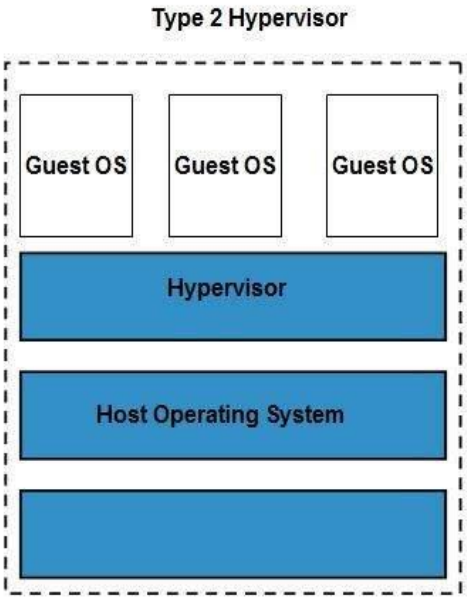
Υπενθυμίζεται στο σημείο αυτό ότι ένας hypervisor επιτρέπει σε αρκετές εικονικές μηχανές να διαμοιράζονται ένα σύστημα. Η στοίβα του λογισμικού μπορεί να οργανωθεί με διάφορους τρόπους [M2013]:

1. Κλασσική οργάνωση (1ος τύπος hypervisor): ο hypervisor είναι ένα επίπεδο λογισμικού που εκτελείται απ' ευθείας πάνω στο υλικό της μηχανής. Το βασικό της πλεονέκτημα είναι η απόδοση. Παραδείγματα: VMware ESX, ESXi Servers, Xen, OS370, LynxSecure, RTS Hypervisor, Oracle VM, Sun xVM Server, VirtualLogic VLX και Denali (Εικόνα 25).

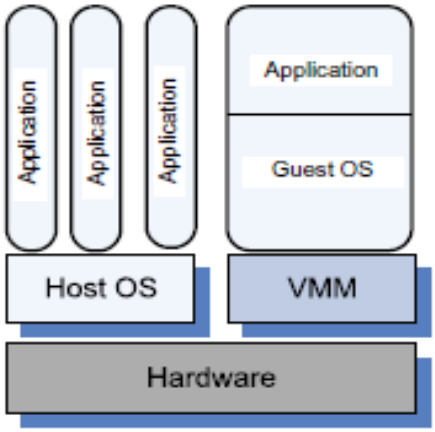


Εικόνα 25: Hypervisor (τύπος 1).

2. Hosted (2ος τύπος hypervisor): ο hypervisor εκτελείται στην κορυφή του υπάρχοντος λειτουργικού συστήματος. Ο hypervisor είναι ένα λογισμικό διεπαφής που εξομοιώνει τις συσκευές με τις οποίες το σύστημα αλληλεπιδρά (Εικόνα 26). Το βασικό πλεονέκτημα αυτής της λύσης είναι ότι η εικονική μηχανή είναι πιο εύκολο να εγκατασταθεί. Ένα άλλο πλεονέκτημα αυτής της οργάνωσης είναι ότι ο hypervisor μπορεί να χρησιμοποιήσει αρκετά στοιχεία του υπάρχοντος λειτουργικού συστήματος, όπως τον χρονοπρογραμματιστή, τον σελιδοποιητή, τους οδηγούς λειτουργιών Ε/Ε, αντί να παρέχει τα δικά του. Από την άλλη όμως η απλότητα αυτή είναι η αυξημένη επιβάρυνση και πτώση στην απόδοση, κι αυτό γιατί οι λειτουργίες Ε/Ε, τα σφάλματα σελίδας και οι αιτήσεις χρονοπρογραμματισμού ενός φιλοξενούμενου λειτουργικού συστήματος δεν είναι διαχειριζόμενες απ' ευθείας από τον hypervisor, αλλά δίνονται για εξυπηρέτηση από το υπάρχον λειτουργικό σύστημα. Η απόδοση αλλά και οι προκλήσεις για την υποστήριξη πλήρους απομόνωσης των εικονικών μηχανών κάνουν αυτή τη λύση λιγότερο ελκυστική για εξυπηρετητές σε περιβάλλοντα νέφους. Παράδειγμα: user-mode Linux, Containers, KVM, Microsoft Hyper V, VMWare Fusion, Virtual Server 2005 R2, Windows Virtual PC and VMWare workstation 6.0.
3. Υβριδική οργάνωση (3ος τύπος hypervisor): ο hypervisor διαμοιράζεται το υλικό με το υπάρχον λειτουργικό σύστημα (Εικόνα 27). Παραδείγματα: VMware Workstation.



Εικόνα 26: Hypervisor (τύπος 2).

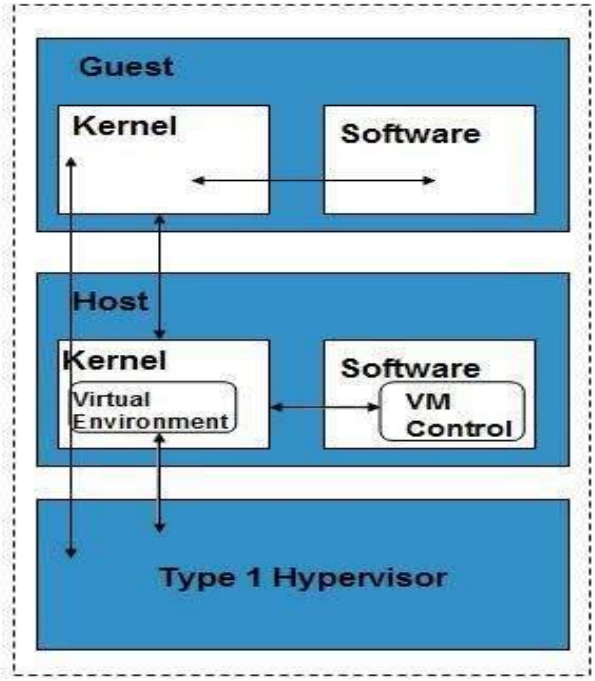


Εικόνα 27: Hypervisor (τύπος 3).

Υπάρχουν τρεις τύποι εικονικοποίησης του υλικού:

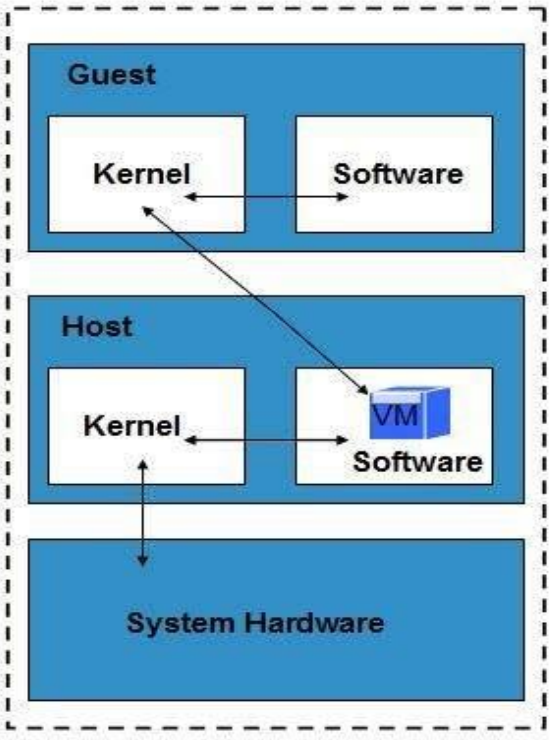
- Πλήρης εικονικοποίηση,
- Εξομείωση
- Paravirtualization

Στην πλήρη εικονικοποίηση το βασικό υλικό του συστήματος προσομοιώνεται πλήρως. Κάθε φιλοξενούμενο λογισμικό δεν χρειάζεται καμιά τροποποίηση για να εκτελεστεί (Εικόνα 28). Παράδειγμα: VMware.



Εικόνα 28: Πλήρης εικονικοποίηση.

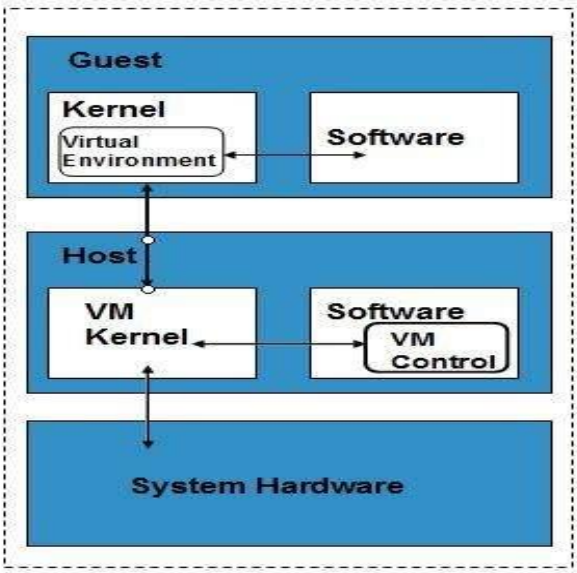
Στην εξομοίωση η εικονική μηχανή προσομοιώνει το υλικό και έτσι γίνεται ανεξάρτητο από αυτή. Έτσι το φιλοξενούμενο λειτουργικό σύστημα δεν χρειάζεται καμιά τροποποίηση (Εικόνα 29).



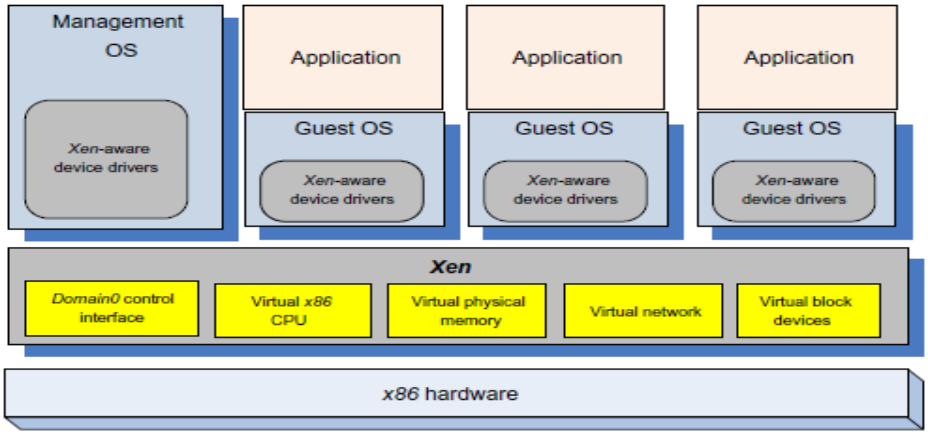
Εικόνα 29: Εξομοίωση.

Σύμφωνα με τη μέθοδο paravirtualization, το υλικό δεν προσομοιώνεται όπως ακριβώς είναι. Συγκεκριμένα, το φιλοξενούμενο λειτουργικό σύστημα τροποποιείται ώστε να χρησιμοποιεί μόνο εντολές που μπορούν να εικονικοποιηθούν. Κάθε εικονική μηχανή εκτελείται σε ένα ελαφρά τροποποιημένο αντίγραφο του πραγματικού υλικού (Εικόνα 30). Οι λόγοι η τεχνική αυτή υιοθετείται συχνά είναι α) κάποια στοιχεία του υλικού δεν μπορούν να εικονικοποιηθούν, β) για τη βελτίωση της απόδοσης και γ) για τη δημιουργία μιας απλούστερης διεπαφής. Παράδειγμα: Xen, Denali.

Το φιλοξενούμενο λογισμικό εκτελείται σε απομονωμένο domain. Χαρακτηριστικό παράδειγμα είναι το Xen, που χρησιμοποιεί τον όρο Dom για να αναφερθεί στους ξεχωριστούς χώρους διευθύνσεων όπου κατοικούν το φιλοξενούμενο λειτουργικό σύστημα και οι εφαρμογές που εκτελούνται υπό αυτό το φιλοξενούμενο λειτουργικό σύστημα. Κάθε ξεχωριστό domain εκτελείται σε μία ξεχωριστή εικονική κεντρική μονάδα επεξεργασίας. Το Dom_0 είναι αφιερωμένο στην εκτέλεση των συναρτήσεων ελέγχου και των προνομιακών εντολών του Xen το Dom_U είναι το domain χρήστη (Εικόνα 31).



Εικόνα 30: Paravirtualization.



Εικόνα 31: Η αρχιτεκτονική του Xen.

2.7 Ασφάλεια

Η ασφάλεια είναι μέγιστης σημασίας στα συστήματα νέφους. Τα δεδομένα πρέπει να αποθηκεύονται σε κρυπτογραφημένη μορφή. Πριν την ενσωμάτωση ενός πόρου σε ένα σύστημα νέφους, χρειάζεται να γίνει ανάλυση κάποιων στοιχείων του πόρου, όπως:

- Ανάλυση της «ευαισθησίας σε ρίσκο» για τον πόρο που έχει επιλεγεί να ενταχθεί στο νέφος.

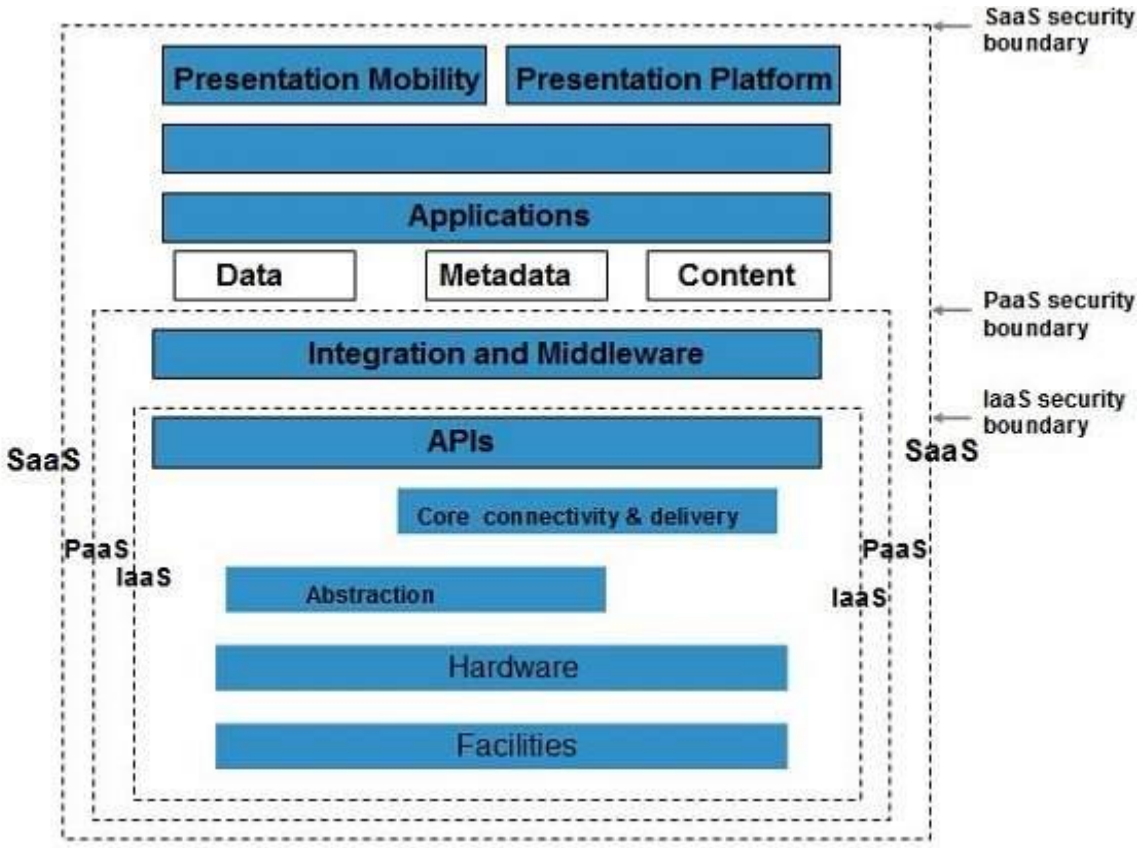
- Θεωρούμε κάποια μοντέλα νέφους όπως IaaS, PaaS, SaaS. Τα μοντέλα αυτά απαιτούν ο πελάτης να είναι υπεύθυνος για την ασφάλεια σε διαφορετικά επίπεδα της εξυπηρέτησης.
- Κατανόηση του μηχανισμού του παρόχου για την αποθήκευση και τη μεταφορά δεδομένων μέσα και έξω από το νέφος.

Το ρίσκο γενικά εξαρτάται από το μοντέλο και τον τύπο του νέφους.

Ένα συγκεκριμένο μοντέλο ορίζει τα όρια των ευθυνών μεταξύ του παρόχου και του πελάτη. Το μοντέλο Cloud Security Aliance (CSA) ορίζει τα όρια μεταξύ κάθε μοντέλου νέφους και δείχνει πως διαφορετικές λειτουργικές μονάδες σχετίζονται η μια με την άλλη (Εικόνα 32). Από την εικόνα αυτή διαφαίνεται ότι:

- Η IaaS είναι ένα μοντέλο στοιχειώδους επιπέδου με την PaaS και την SaaS στα δύο παραπάνω επίπεδα υπηρεσίας.
- Ελέγχοντας τη στοίβα από κάτω προς τα πάνω, κάθε υπηρεσία κληρονομεί δυνατότητες και θέματα ασφάλειας από τα από κάτω μοντέλα.
- Η IaaS έχει το κατώτερο επίπεδο ολοκλήρωσης λειτουργιών και ασφάλειας, ενώ η SaaS έχει το ανώτερο.
- Κάθε μηχανισμός ασφάλειας χαμηλότερα από το όριο ασφάλειας πρέπει να κατασκευαστεί στο σύστημα και να συντηρηθεί από τον πελάτη.

Παρόλο που κάθε μοντέλο νέφους διαθέτει μηχανισμούς ασφάλειας, οι ανάγκες ασφάλειας εξαρτώνται ακόμη από το αν τοποθετούνται οι υπηρεσίες σε δημόσιο ή ιδιωτικό ή υβριδικό, κλπ. νέφος.



Εικόνα 32: Cloud Security Aliance model.

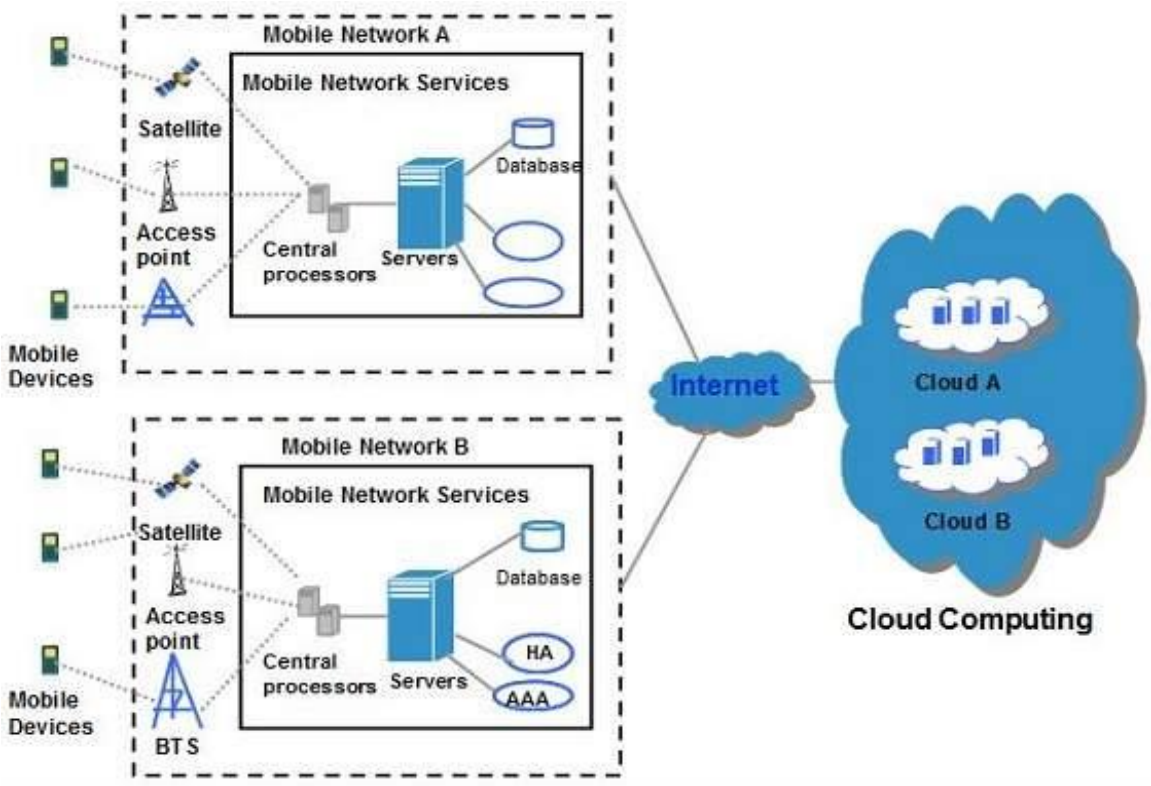
2.8 Κινητός Υπολογισμός σε Νέφος

Για τα έξυπνα τηλέφωνα που έχουν μεγάλη ανάγκη από Internet μέσα, ο υπολογισμός νέφους προσφέρει λιγότερη επεξεργασία και λιγότερη κατανάλωση ισχύος. Έτσι, η επεξεργασία γίνεται στο νέφος, η αποθήκευση γίνεται στο νέφος, και οι κινητές συσκευές χρησιμοποιούνται απλά μόνο για την απεικόνιση.

Τα σημερινά έξυπνα τηλέφωνα διαθέτουν πολλές υπηρεσίες νέφους διασυνδέοντας υπηρεσίες που χρησιμοποιούν υπηρεσίες ιστού. Αυτές οι υπηρεσίες ιστού αναπτύσσονται στο νέφος. Υπάρχουν αρκετά διαθέσιμα λειτουργικά συστήματα για έξυπνα τηλέφωνα, όπως το Android της Google, της Apple το iOS, RIM Blackberry, Symbian, και Windows Mobile Phone. Κάθε μία από αυτές τις πλατφόρμες υποστηρίζει εφαρμογές τρίτης έμπιστης οντότητας (third party) που υιοθετούνται στο νέφος.

Η αρχιτεκτονική ενός τέτοιου συστήματος που υποστηρίζει κινητό υπολογισμό νέφους φαίνεται στην Εικόνα 33. Μια τέτοια αρχιτεκτονική περιλαμβάνει τέσσερις τύπους πόρους νέφους:

- Μακρινό κινητό νέφος
- Μακρινό μη κινητό νέφος
- Κοντινές οντότητες κινητού υπολογισμού
- Κοντινές οντότητες μη κινητού υπολογισμού



Εικόνα 33: Αρχιτεκτονική κινητού υπολογισμού σε νέφος.

2.9 Αρχιτεκτονική Κατανεμημένων Εφαρμογών

Καθώς η χρήση του παγκόσμιου ιστού για επιχειρηματικές εφαρμογές ξεκίνησε στα 1995, μια κατανεμημένη αρχιτεκτονική εφαρμογών (που πολλές φορές αναφέρεται σαν αρχιτεκτονική N επιπέδων – N tier) ήταν επείγουσα. Αυτή η αρχιτεκτονική έμοιαζε διαφορετική από την παραδοσιακή client – server αρχιτεκτονική, όπου η βάση δεδομένων γενικά ήταν εγκατεστημένη σε ένα κεντρικό εξυπηρετητή και ο «βαρύς» πελάτης που περιελάμβανε τη διεπαφή χρήστη και τη λογική της εφαρμογής ήταν εγκατεστημένος στην επιφάνεια εργασίας του κάθε χρήστη. Υπήρχαν διάφορα ελαττώματα σε αυτό το μοντέλο, συμπεριλαμβανομένων θεμάτων κλιμάκωσης λόγω των αφιερωμένων συνδέσεων της βάσης με τον κάθε χρήστη και θεμάτων που έχουν να

κάνουν με τη λειτουργική επιβάρυνση της διανομής, εγκατάστασης και συντήρησης του λογισμικού στη μεριά του πελάτη.

Η αρχιτεκτονική κατανεμημένης αρχιτεκτονικής έγινε δυνατή εξ αιτίας δύο πραγμάτων. Πρώτον, ο φυλλομετρητής web ήταν επιτακτικό να γίνει ένα πανταχού παρόν εργαλείο στην επιφάνεια εργασίας. Δεύτερον, για ένα κομμάτι λογισμικού γνωστό σαν εξυπηρετητής εφαρμογών, ήταν επιτακτική ανάγκη να μετατοπιστεί στη μεριά του πελάτη μέσω του Web browser για λόγους αποσυμφόρησης.

Επίπεδο Χρήστη

Το επίπεδο χρήστη αποτελείται από τον web browser που μπορεί αν δεν είναι εγκατεστημένος ήδη, να αποκτηθεί και εγκατασταθεί ελεύθερα από το Internet. Εναλλακτικά μπορεί να υπάρχουν εφαρμογές πελάτη, αντί του web browser. Αυτές οι εφαρμογές συνήθως διαχειρίζονται απλά την είσοδο και την έξοδο του εξυπηρετητή και μπορούν εύκολα να μεταφορτωθούν αν και όποτε χρειαστούν.

Επίπεδο Εφαρμογής

Το επίπεδο εφαρμογής περιλαμβάνει τον εξυπηρετητή εφαρμογών ο οποίος χειρίζεται τις HTTP/HTTPS αιτήσεις σε συνεργασία με το επίπεδο αποθήκευσης δεδομένων. Το επίπεδο αυτό γενικά παρέχει υπηρεσίες όπως ομαδοποίηση (clustering) και εξισορρόπηση φορτίου μεταξύ των εξυπηρετητών για λόγους κλιμάκωσης και προσαρμοστικότητας, διαχείρισης των δεδομένων των συνόδων των χρηστών, διαχείρισης των συνδέσεων της βάσης δεδομένων και διαχείρισης των αντικειμένων των εφαρμογών που υλοποιούν την εφαρμογή.

Επίπεδο πρόσβασης στα Δεδομένα

Το επίπεδο αυτό διαχειρίζεται την επεξεργασία όλων των δεδομένων που ζητούνται από τις αιτήσεις των χρηστών, είτε από μια δομημένη τύπου SQL βάση δεδομένων, είτε από ένα αδόμητο σύστημα αρχείων. Το επίπεδο αυτό γενικά παρέχει ομαδοποίηση για λόγους κλιμάκωσης και προσαρμοστικότητας. Πέρα από το διαχωρισμό ανάμεσα στη λογική της υπηρεσίας και τα δεδομένα, υπάρχουν και άλλοι λόγοι για το διαχωρισμό του επιπέδου αυτού από το επίπεδο εφαρμογής. Τέτοιοι λόγοι είναι οι διαφορετικές απαιτήσεις για το υλικό (π.χ. βέλτιστη πρόσβαση στο δίσκο για ένα εξυπηρετητή βάσεων δεδομένων έναντι πρόσθετης RAM που απαιτείται από τον εξυπηρετητή εφαρμογών), και πρόσθετη ασφάλεια (διαχωρισμός του επιπέδου αυτού σε ένα εσωτερικό τμήμα δικτύου για τη μείωση του ρίσκου μη εξουσιοδοτημένης πρόσβασης σε κρίσιμα δεδομένα).

Επίπεδο Αποθήκευσης Δεδομένων

Το επίπεδο αυτό χρησιμοποιείται για τη φυσική αποθήκευση των δεδομένων. Σε μερικές περιπτώσεις το επίπεδο αυτό φιλοξενείται στον ίδιο φυσικό εξυπηρετητή με το επίπεδο

πρόσβασης στα δεδομένα. Σε άλλες περιπτώσεις χρησιμοποιείται ένα Δίκτυο Περιοχής Αποθήκευσης Δεδομένων (Storage Area Network) για βέλτιστη αποθήκευση και προσαρμοστικότητα.

2.10 Αρχιτεκτονική Λογισμικού Σαν Υπηρεσία (SaaS Architecture)

Το μοντέλο Λογισμικού ως Υπηρεσία περικλείει την ιδέα της αρχιτεκτονικής κατανεμημένων εφαρμογών, αλλά επιπλέον επεκτείνει την αρχιτεκτονική για να συμπεριλάβει στοιχεία για την διευκόλυνση και την βελτίωση του μοντέλου της επιχείρησης. Ένας κλασσικός κατασκευαστής λογισμικού πρωταρχικά απασχολείται με τη λειτουργικότητα της εφαρμογής και οι πελάτες είναι υπεύθυνοι για τη λειτουργία και τη διαχείριση του περιβάλλοντος στο οποίο εκτελείται το λογισμικό. Από την άλλη, ένας κατασκευαστής SaaS απασχολείται και με τη λειτουργία και τη διαχείριση του περιβάλλοντος που υποστηρίζει όλους τους πελάτες του.

Ένα πρόσθετο επίπεδο, που ονομάζεται επίπεδο κατανομής, εισάγεται για την αναγνώριση ότι αιτήσεις εξυπηρέτησης χρειάζεται να δρομολογούνται ανάμεσα σε περισσότερα από ένα φυσικά λειτουργικά περιβάλλοντα για πολλούς λόγους. Πρόσθετα στοιχεία επιπέδου εφαρμογής αναγνωρίζονται, που λαμβάνουν υπόψη την πρόσθετη λειτουργικότητα μια ώριμη SaaS παροχή μπορεί να απαιτεί. Τα ζητήματα διαχείρισης και επίβλεψης παρουσιάζονται σαν στοιχεία που λειτουργούν μεταξύ των επιπέδων. Αυτά τα στοιχεία είναι κρίσιμα για την αποδοτική λειτουργία μιας επιχείρησης SaaS.

Επίπεδο Κατανομής (Distribution tier)

Το επίπεδο κατανομής εισάγεται στην αρχιτεκτονική του Λογισμικού Σαν Υπηρεσία για να επιτύχει την εξισορρόπηση φόρτου. Ο όρος εξισορρόπηση φόρτου γενικά σχετίζεται με τεχνικές για διασπορά εργασιών με σκοπό την βελτιστοποίηση της χρησιμοποίησης των πόρων, της ρυθμοαπόδοσης (throughput) και του χρόνου απόκρισης. Για μία επιχείρηση που παρέχει SaaS είναι σίγουρο πως στους πελάτες πρέπει να παρέχεται το υψηλότερο επίπεδο υπηρεσίας. Όμως, υπάρχουν άλλοι λόγοι που αιτήσεις για εξυπηρέτηση ίσως χρειάζεται να δρομολογηθούν σε άλλα φυσικά λειτουργικά περιβάλλοντα.

Ένας λόγος είναι η αλληλεπίδραση πολλαπλών, γεωγραφικά ανόμοιων κέντρων αποθήκευσης δεδομένων (data centers) για λόγους προσαρμοστικότητας. Η ιδέα είναι κάθε διαφορετικός πελάτης να δρομολογείται σε διαφορετικό βασικό site. Όλα τα sites χρησιμοποιούνται για την εξυπηρέτηση αιτήσεων και λειτουργούν ως εναλλακτικά sites το ένα για το άλλο, χρησιμοποιώντας έτσι με αποδοτικότητα τους πόρους και παρέχοντας υψηλή διαθεσιμότητα.

Ένας άλλος λόγος συγκεκριμένων αιτήσεων εξυπηρέτησης πελατών σε ειδικά περιβάλλοντα είναι για να είναι υπάρχει συμβατότητα με διεθνείς νόμους ιδιωτικότητας

των δεδομένων. Γεγονός είναι ότι πρόκειται για ένα μεγάλο θέμα και πολλές χώρες έχουν εφαρμόσει πολύ διαφορετικές προσεγγίσεις για την προστασία των δεδομένων των πολιτών τους. Επίσης πολλοί πελάτες από συγκεκριμένες χώρες απαιτούν τα δεδομένα τους να αποθηκεύονται εντός της χώρας τους. Εάν ένας SaaS κατασκευαστής θέλει να αναπτύξει την επιχείρησή του σε αυτές τις χώρες, μπορεί να απαιτηθούν ανόμοια διεθνή περιβάλλοντα παραγωγής και η δρομολόγηση αιτήσεων να είναι υποχρεωτική.

Άλλα συστατικά επιπέδου εφαρμογής

Το μοντέλο της SaaS αρχιτεκτονικής εισάγει κάποια λεπτομέρεια στο επίπεδο της εφαρμογής. Σε αντίθεση με ένα εξυπηρετητή γενικού σκοπού που εκτελεί μια εφαρμογή, υπάρχουν διαφορετικοί τύποι ειδικών εξυπηρετητών που παρέχουν ειδικούς τύπους λειτουργικότητας.

Εξυπηρετητής διαχείρισης ταυτοτήτων

Η ασφάλεια είναι ένα σημαντικό θέμα σε ένα περιβάλλον SaaS. Έχοντας τη δυνατότητα για ευέλικτο χειρισμό της διαχείρισης των ταυτοτήτων με τυποποιημένο τρόπο, μπορεί να επιτραπεί στους πελάτες να επιλέξουν μεταξύ πολλαπλών τρόπων πιστοποίησης. Ένα παράδειγμα εξυπηρετητή διαχείρισης ταυτοτήτων είναι το OpenSSO. Πρόκειται για λογισμικό ανοικτού κώδικα που παρέχει ένα εξυπηρετητή διαχείρισης πρόσβασης επιπέδου επιχείρησης που βασίζεται στης Sun Java System Access Manager και θα είναι ο πυρήνας των επερχόμενων εκδόσεων του Access Manager και Federation Manager της Sun.

Εξυπηρετητής ενοποίησης

Όπως αναφέρθηκε στο μοντέλο εξέλιξης των τεσσάρων βημάτων, η ενοποίηση είναι κεντρικό θέμα καθώς το μοντέλο SaaS ωριμάζει. Παρέχοντας ένα ειδικό εξυπηρετητή ενοποίησης για ολοκλήρωση με βιομηχανικές πλατφόρμες όπως HR, CRM, κλπ., θα είναι πιο εύκολο για τους πελάτες να ενσωματώσουν μια SaaS λύση στις ήδη υπάρχουσες πλατφόρμες τους.

Εξυπηρετητής επικοινωνιών

Τόσο οι εξερχόμενες όσο και οι εισερχόμενες επικοινωνίες είναι πρόσθετα συστατικά μιας SaaS παροχής που συχνά είναι απαραίτητα, ενώ μερικές φορές παραβλέπονται. Κατ' ελάχιστο είναι χρήσιμο να υπάρχει δυνατότητα επικοινωνίας με τους χρήστες (π.χ. μέσω ηλεκτρονικού ταχυδρομείου και απλών μηνυμάτων κειμένου) για να τους γίνει γνωστό ότι υπάρχει κάποια εργασία να κάνουν στην SaaS εφαρμογή. Ένας εξυπηρετητής ηλεκτρονικού ταχυδρομείου, γνωστός ως Mail Transport Agent – MTA, είναι μια απλή

διεργασία που εκτελείται, αλλά η παροχή μιας εύρωστης υποδομής ηλεκτρονικού ταχυδρομείου απαιτεί περισσότερη σκέψη και σχέδιο. Επιπλέον, το SMTP δεν είναι απαραίτητα το μόνο πρωτόκολλο επικοινωνίας που μπορεί να υποστηριχθεί, έτσι η σκέψη για ένα εξυπηρετητή επικοινωνιών σαν ένα πιο γενικό συστατικό στην αρχιτεκτονική είναι χρήσιμη.

3. Βιβλιογραφικές Αναφορές

[AFGJKKLPRSZ2009] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Kowinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", Technical Report No. UCB/EECS-2009-28, 2009.

[AL1998] Yariv Aridor and Danny B. Lange, "Agent Design Patterns: Elements of Agent Application Design", In proceeding of the 2th ACM international conference on Autonomous Agents (Agents '98), p. 108-115, 1998.

[AS2004] S. Androutsellis and D. Spinellis, "A Survey or Peer-to-Peer Content Distribution Technologies", ACM Computing Surveys, vol. 36, pp. 335-371, 2004.

[BGKMSZ2002] Bernstein, P., F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini and I. Zaihrayeu, "Data management for peer-to-peer computing: A vision", In Proceedings of the Workshop on the Web and Databases, 2002.

[CKK2000] Chen, Y., R. Katz and J. Kubiatowicz, "Scan: A dynamic, scalable and efficient content distribution network", In Proceeding of International Conference on Pervasive Computing, 2000.

[DFM2000] Dingledine, R., M. Freedman and D. Molnar, "The FreeHaven project: Distributed anonymous storage service", In Workshop on Design Issues in Anonymity and Unobservability, pp. 67-95, 2000

[F2003] FastTrack Accessed on-line, 2003 (www.fasttrack.nu)

[FIPS1995] FIPS 180-1. Secure Hash Standard. U.S. Department of Commerce/ NIST, National Technical Information Service, Springfield, VA, 1995.

[G2003] Gnutella, 2003 (www.gnutella.wego.com)

[HIMT2003] Halevy, A., Z. Ives, P. Mork and I. Tatarinov, "Piazza: Data management infrastructure for semantic web applications", In Proceeding of the 12th International Conference on World Wide Web, Budapest, Hungary, 2003.

[HR2002] Hand, S. and T. Roscoe, "Mnemosyne: Peer-to-peer steganographic storage", In Proceeding of the 1st International Workshop on Peer-to-Peer Systems, MIT Faculty Club, Cambridge, MA, 2002.

[HHLL2003] Huebsch, R., J. Hellerstein, N. Lanham and B. Thau Loo, "Querying the internet with pier", In Proceeding of the 29th VLDB Conference, Berlin, Germany, 2003.

[J2001] Jovanovich, M., F. Annexstein and K. Berman, Scalability issues in large peer-to-peer networks- a case study of Gnutella, ECECES Department, University of Cincinnati, Cincinnati, 2001.

[J200] Jovanovich, M., Modeling large-scale peer-to-peer networks and a case study of Gnutella, Department of Electical and Computer Engineering and Computer Science, University of Cincinnati, Cincinnati, 2000.

[KLLLLP1997] Karger, D., E. Lehman, F. Leighton, M. Levine, D. Lewin and R. Panigrhy, "Consistent hashing and random trees: Distributed cashing protocols for relieving hot spots on the world wide web", In Proceeding of the 29th Annual ACM Symposium on Theory of Computing, pp.654-663, 1997.

Kazaa, 2003 (www.kazaa.com)

[KBCEGGWWZ2000] Kubiatowicz, J., D. Bindel, Y. Chen, P. Eaton, D. Geels, S. Gummadi, H. Weimmer, C. Wells and B. Zahao, "Oceanstore: An architecture for global-scale persistent storage", In Proceeding of ACM ASPLOS, 2000.

[LMFS2003] Emerson Ferreira de Araújo Lima, Patrícia Duarte de Lima Machado, Jorge César Abrantes de Figueiredo, and Flávio Ronison Sampaio, "Implementing Mobile Agent Design Patterns in the JADE framework", EXP in Search of Innovation,Vol.-3, No.-3, September 2003.

[LRS2002] Lv, Q., S. Ratnasamy and S. Shenker, "Can heterogeneity make Gnutella scalable?", In Proceeding of the 1st International Workshop on Peer-to-Peer Systems, MIT Faculty Club, Cambridge, MA, 2002

[M2013] Marinescu D, "Cloud Computing: Theory and Practice", Morgan Kaufmann, ISBN: 978-0-12404-627-6, 2013.

MojoNation, 2003 (www.mojonation.net)

[NWDSNN2003] Nejdl, W., B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilson, M. Palmer and T. Risch, "Edutella: A p2p networking infrastructure based on rdf", In Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, 2003.

[RR1997] C., R. Rajaraman and A. Richa, "Accessing nearby copies of replicated objects in a distributed enviroment", In Proceeding of ACM SPAA, 1997.

[RFHK2001] Ratnasamy, S., P. Francis, M. Handley and R. Karp, "A scalable content-addressable network", In Proceeding of SIGCOMM, 2001.

[RW2001] Rhea, S., C. Wells et al., Maintenance- free global storage, pp. 40-49, 2001.

[RF2002] Ripeanu, M. and I. Foster, "Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems", In Proceeding of the 1st International Workshop on Peer-to-Peer Systems, MIT Faculty Club, Cambridge, MA, 2002.

[SGG2002] Saroiu, S., P. Gummadi and S. Gribble, "Exploring the design space of distributed peer-to-peer systems: Comparing the web, TRIAD and Chord/CFS", In Proceeding of the the 1st International Workshop on Peer-to-Peer Systems, MIT Faculty Club, Cambridge, MA, 2002.

[SG1995] Shaw, M. and D. Garlan, In Computer Science Today: Recent Trends and Developments, Lecture Nodes in Computer Science, 1995.

[S2000] Shirky, C., What is peer-to-peer and what isn't, 2000 (www.oreillynet.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html).

[SW2004] Steinmetz, R. and K. Wehrle, Peer-to-peer networking & computing, Informatik-Spektrum, 2004.

[SMKKB2001] Stoica, I., R. Morris, D. Karger, M. Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", In Proceeding of SIGCOMM, 2001.

[WMB1999] Witten, I., A. Moffat and T. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd ed. Morgan Kauffman, 1999.

[ZKJ2001] Zhao, B., J. Kubiatowicz and A. Joseph, An infrastructure for fault-tolerant wide-area location and routing, Computer Science Division, University of California, Berkeley, 2001.

[ZMK2002] Zhichen, X., M. Mahalingam and M. Karlsson, Turning heterogeneity to an advantage in overlay routing, 2002.

[cloud] http://www.tutorialspoint.com/cloud_computing/index.htm