ΙΕΡΑΡΧΙΕΣ ΚΟΜΒΩΝ ΔΟΜΗΜΕΝΕΣ ΣΕ ΔΑΚΤΥΛΙΟ ΓΙΑ P2P ΣΥΣΤΗΜΑΤΑ ΒΑΣΙΣΜΕΝΑ ΣΕ RDF ΣΧΗΜΑΤΑ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Νικόλαο Κρεμμυδά

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Απρίλιος 2007

ΑΦΙΕΡΩΣΗ

Στους γονείς μου, Βάσω και Γιώργο…

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω την καθηγήτριά μου και ερευνητική σύμβουλό μου Ευαγγελία Πιτουρά, για τις συμβουλές και τις υποδείξεις της καθ' όλη τη διάρκεια της παρούσας εργασίας. Ακόμη, θα ήθελα να ευχαριστήσω τους κύριους Βασιλειάδη και Ζάρρα, για τις τελικές διορθώσεις της παρούσας διατριβής. Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου, για την αμέριστη ηθική και οικονομική συμπαράσταση που μου παρείχε σε όλη τη διάρκεια των σπουδών μου. Τέλος, θα ήθελα να ευχαριστήσω και όλους τους φίλους και συναδέλφους μου, των οποίων η βοήθεια, οι γνώσεις, αλλά πάνω από όλα η φιλία τους, υπήρξε καθοριστικός παράγοντας για τη διεκπεραίωση των μεταπτυχιακών σπουδών μου, στα Ιωάννινα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Νικόλαος Κρεμμυδάς του Γεωργίου και της Βασιλικής. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Μάρτιος, 2007. Τίτλος Διατριβής. Επιβλέπουσα: Ευαγγελία Πιτουρά.

Τα P2P δίκτυα είναι κατανεμημένα συστήματα ομότιμων κόμβων (peers). Κάθε κόμβος σε αυτά τα συστήματα έχει περίπου ίδιες δυνατότητες, κρατάει ίδιο μέγεθος πληροφορίας και έχει τις ίδιες ευθύνες με κάθε άλλο κόμβο στο σύστημα. Τα δίκτυα αυτά, χρησιμοποιούνται ευρέως, επειδή είναι ανεκτικά σε σφάλματα, μπορούν να κλιμακωθούν σε μεγάλα πλήθη κόμβων και παρουσιάζουν (τα περισσότερα) ικανοποιητικούς χρόνους σε ότι αφορά την εισαγωγή/διαγραφή κόμβων σε αυτά και την αναζήτηση ερωτημάτων. Σε ένα τέτοιο δίκτυο P2P μελετάμε την διαχείριση RDF Σχημάτων. Τα RDF Σχήματα είναι σημασιολογικοί γράφοι οι οποίοι μπορούν να χρησιμοποιηθούν από τους κόμβους του δικτύου για να περιγράψουν τις βάσεις μετά-δεδομένων τους με έναν εννοιολογικό τρόπο. Η ανάγκη για αποδοτική διαχείριση μετά-δεδομένων, έχει επιφέρει την ανάπτυξη ενός σημαντικού αριθμού μοντέλων τα οποία διαχειρίζονται μετά-δεδομένα σε δίκτυα P2P. Ωστόσο, λίγη είναι η δουλειά που έχει γίνει για την δημιουργία ενός σημασιολογικού δικτύου το οποίο θα διαχειρίζεται σχήματα βάσεων μετά-δεδομένων.

Στην παρούσα διατριβή ερευνούμε την αποδοτική διαχείριση RDF (Resource Description Framework) Σχημάτων σε ένα δίκτυο P2P (peer-to-peer). Προσπαθούμε, δηλαδή, να επιτύχουμε την αποδοτική εισαγωγή/διαγραφή κόμβων σε ένα P2P δίκτυο, στο οποίο κάθε κόμβος αναπαριστά τη βάση μετά-δεδομένων του με ένα RDF Σχήμα. Επίσης, ενδιαφερόμαστε για την αποδοτική αναζήτηση RDF Σχημάτων σε ένα τέτοιο δίκτυο. Πιο αναλυτικά, το σύστημα που περιγράφουμε, βασίζεται σε μια δομή δακτυλίου και σε διαφορετικές ιεραρχίες από RDF Σχήματα. Προσπαθούμε να δομήσουμε κόμβους με σχήμα, το οποίο είναι υποσύνολο του ίδιου RDF Σχήματος,

στην ίδια ιεραρχία, ενώ ο ριζικός κόμβος κάθε ιεραρχίας τοποθετείται σε μια θέση στον δακτύλιο. Επίσης, περιγράφουμε κι έναν τρόπο εισαγωγής/διαγραφής ενός σχήματος σε αυτό το σύστημα, καθώς κι έναν τρόπο αναζήτησης ενός ερωτήματος. Τέλος, εφαρμόζουμε μια τεχνική εξισορρόπησης φόρτου για το δίκτυό μας. Στα πειράματά μας, μετράμε την απόδοση του συστήματός μας για διαφορετικές περιπτώσεις δικτύων. Πρώτα για ένα δίκτυο κόμβων το οποίο έχει πολλές ιεραρχίες και λίγους κόμβους σε κάθε ιεραρχία, ύστερα, για ένα δίκτυο το οποίο έχει λίγες ιεραρχίες και πολλούς κόμβους σε κάθε ιεραρχία και για ένα δίκτυο στο οποίο το πλήθος των ιεραρχιών και των κόμβων σε κάθε ιεραρχία είναι τυχαίο. Επίσης, συγκρίνουμε το σύστημά μας με ένα σύστημα στο οποίο οι κόμβοι συνδέονται μεταξύ τους τυχαία (αδόμητο σύστημα), καθώς και με ένα σύστημα από την υπάρχουσα βιβλιογραφία. Το σύστημά μας δουλεύει πιο αποδοτικά όταν το σύνολο των ιεραρχιών είναι μικρό σε σχέση με τον συνολικό αριθμό των κόμβων, ενώ, όταν συμβαίνει το αντίθετο, παρατηρούμε μια μεγάλη αύξηση στον αριθμό των μηνυμάτων που απαιτούνται για τη εισαγωγή/διαγραφή κόμβου και την αναζήτηση ενός ερωτήματος.

EXTENDED ABSTRACT IN ENGLISH

The need for storing and representing metadata has resulted in a large number of models, concerning the efficient storage and search of metadata, in various systems. RDF (Resource Description Framework) Schemas have been the main model for the representation of metadata, which allow us to represent data with a specific form. RDF Schemas are semantic graphs that allow nodes to represent the content of their metadata bases, semantically.

An RDF Schema consists of classes which are linked by attributes. Each class is a conceptual representation of a set of metadata of similar meaning. In this way, if a node wants to group some metadata of its metadata base, it can include them in specific classes, as instances of those classes. If every node in a network represents its metadata by an RDF Schema, then it is possible to create a semantic network of metadata, which is very important. In such networks, we can make more sophisticated questions (that can include the semantics of a part of the network) than in simpler networks. Moreover, we would like to find a network of nodes (which publish their metadata bases by RDF Schemas), that can answer questions, concerning RDF Schemas, in little time and with few messages.

The networks that have attracted the most attention are P2P (peer-to-peer) networks (versus the client – server networks). The main characteristic of P2P networks is that all of the nodes are equivalent. That is, all nodes are required to store the same amount of information. Also, all nodes have the same possibilities and obligations with any other node in the network. The most important advantages of P2P networks, (in contradiction to client - server networks), are that P2P networks are scalable (they can support a large number of nodes, efficiently), they are reliable (nodes can fail,

without affecting the rest of the system) and they are efficient (most of the current P2P systems require logarithmic time, in order to answer a question). P2P networks can be distinguished into two groups, depending on the connection of the nodes. The connection can be unstructured (that is, nodes are randomly connected to each other and data are distributed across the network independently of the connection of the nodes) or structured (that is, nodes are connected in a predetermined way, that obey certain rules, and the distribution of data across the network, is based on the topology of nodes).

The problem that concerns us is to find a way to answer complicated questions in P2P networks (for example, we would like to ask for the RDF Schema of a specific part of the network), in little time and with few messages. We expect that a structured system will be much more efficient, regarding the search of RDF data. If, however, we use a structured system, we have to resolve the problem of maintaining the structure, that is, the insertion/deletion of the nodes have to be carried out efficiently. On the other hand, if we use an unstructured P2P network, we no longer have the problem of insertion/deletion of nodes, but the time for answering a complicated query increases dramatically. Each of the systems that have been proposed for managing RDF Schemas presents different problems. In some of them, nodes have to store a large amount of data, while in others, the time for answering a query is linear to the size of the network.

The approach that we present in this thesis is based on the idea of the RDF Schema itself. In an RDF Schema, the classes and the attributes form a hierarchy. Similarly, in our system, nodes are structured semantically, in a hierarchy, based on the RDF Schema of their metadata base. These hierarchies are structured in a ring (similar to the chord ring), so that the search of queries is accomplished in small amount of time. This ring has the characteristics of the Chord ring (*finger tables, successors* etc), something which allows us "to reach" a certain hierarchy in (logn) time, if n is the total number of hierarchies. This way, we achieve an efficient structure for the insertion/deletion of RDF Schemas and for query routing. Our system is compared with an unstructured network of nodes (the nodes are connected randomly) and with a system of the current bibliography. The experiments showed, that the system works

much better (both for the insertion/deletion of a node and for the search of a query), when the number of hierarchies is relatively small, related with the total number of nodes in the network. Also, our system works much better, in relevance with the other two systems with regard to the time of insertion/deletion of nodes and the search of queries. On the other hand, our system needs to contact a lot of nodes in the cases of the node insertion and the query routing, thus, requiring a large number of messages.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Η ανάγκη για αποθήκευση και αναπαράσταση μετά-δεδομένων έχει επιφέρει την ανάπτυξη ενός μεγάλου αριθμού μοντέλων, σχετικά με την αποδοτική αποθήκευση και αναζήτηση μετά-δεδομένων σε διάφορα συστήματα.

Όσον αφορά την αναπαράσταση των μετά-δεδομένων, έχουν επικρατήσει τα RDF (Resource Description Framework) Σχήματα [1], τα οποία μας επιτρέπουν να αναπαριστούμε δεδομένα με συγκεκριμένη μορφή. Τα RDF Σχήματα είναι σημασιολογικοί γράφοι, που επιτρέπουν στους κόμβους ενός δικτύου να αναπαριστούν εννοιολογικά το περιεχόμενο των βάσεων μετά-δεδομένων τους. Ένα RDF Σχήμα αποτελείται από κλάσεις, οι οποίες ενώνονται με ιδιότητες. Κάθε κλάση είναι μια εννοιολογική αναπαράσταση ενός συνόλου μετά-δεδομένων, που έχουν παρόμοια σημασία. Με αυτό τον τρόπο, αν ένας κόμβος θέλει να ομαδοποιήσει τα μετά-δεδομένα της βάσης του, μπορεί να τα εντάξει ως στιγμιότυπα, σε συγκεκριμένες κλάσεις. Αν κάθε κόμβος στο ίδιο δίκτυο, αναπαριστά τα μετά-δεδομένα του με ένα RDF Σχήμα, τότε δημιουργείται ένα σημασιολογικό δίκτυο μετά-δεδομένων, κάτι που είναι πολύ σημαντικό, αφού σε τέτοια δίκτυα μπορούμε να κάνουμε πιο πολύπλοκα ερωτήματα απ' ότι στα απλά δίκτυα. Αυτό που θέλουμε είναι να βρούμε ένα δίκτυο από κόμβους (οι οποίοι δημοσιεύουν τη βάση μετά-δεδομένων τους με RDF Σχήματα), ώστε τα ερωτήματα που θέτουμε και αφορούν RDF Σχήματα, να απαντώνται σε λίγο χρόνο και με όσο το δυνατόν λιγότερα μηνύματα (δηλαδή, να μην απασχολούνται, άσκοπα, κόμβοι που δεν μπορούν να απαντήσουν στα ερωτήματα).

Τα δίκτυα κόμβων που έχουν προσελκύσει την μεγαλύτερη προσοχή, ως μέσα διαμοίρασης δεδομένων είναι τα P2P (peer-to-peer) δίκτυα (σε σχέση με τα δίκτυα client-server). Χαρακτηριστικά παραδείγματα αυτής της προσοχής είναι τα συστήματα Gnutella [2] και Napster [3]. Το χαρακτηριστικό των P2P δικτύων είναι

ότι όλοι οι κόμβοι είναι ισοδύναμοι. Δηλαδή, απαιτείται από αυτούς, να αποθηκεύσουν τον ίδιο όγκο πληροφορίας και να έχουν τις ίδιες δυνατότητες και τις ίδιες υποχρεώσεις με οποιονδήποτε άλλο κόμβο στο δίκτυο. Τα μεγάλα πλεονεκτήματα των P2P δικτύων, σε σχέση με τα client-server δίκτυα, είναι ότι τα πρώτα μπορούν να κλιμακωθούν σε μεγάλο αριθμό κόμβων, είναι αξιόπιστα (μπορεί κάποιοι κόμβοι να «πέσουν», χωρίς να επηρεαστεί το σύστημα) και είναι αποδοτικά (τα περισσότερα απαιτούν λογαριθμικούς χρόνους, στο μέγεθος των κόμβων, για να απαντήσουν σε ένα ερώτημα). Τα P2P δίκτυα χωρίζονται σε δύο μεγάλες ομάδες, ανάλογα με την τοπολογία με την οποία είναι συνδεδεμένοι οι κόμβοι. Αυτή, μπορεί να είναι αδόμητη (δηλαδή οι κόμβοι να είναι τυχαία συνδεδεμένοι και τα δεδομένα να διαμοιράζονται στο δίκτυο, ανεξαρτήτως της τοπολογίας των κόμβων) ή να είναι δομημένη (δηλαδή οι κόμβοι να συνδέονται με καθορισμένο τρόπο, που υπακούει σε κάποιους κανόνες και τα δεδομένα να διαμοιράζονται στο δίκτυο με βάση την τοπολογία των κόμβων).

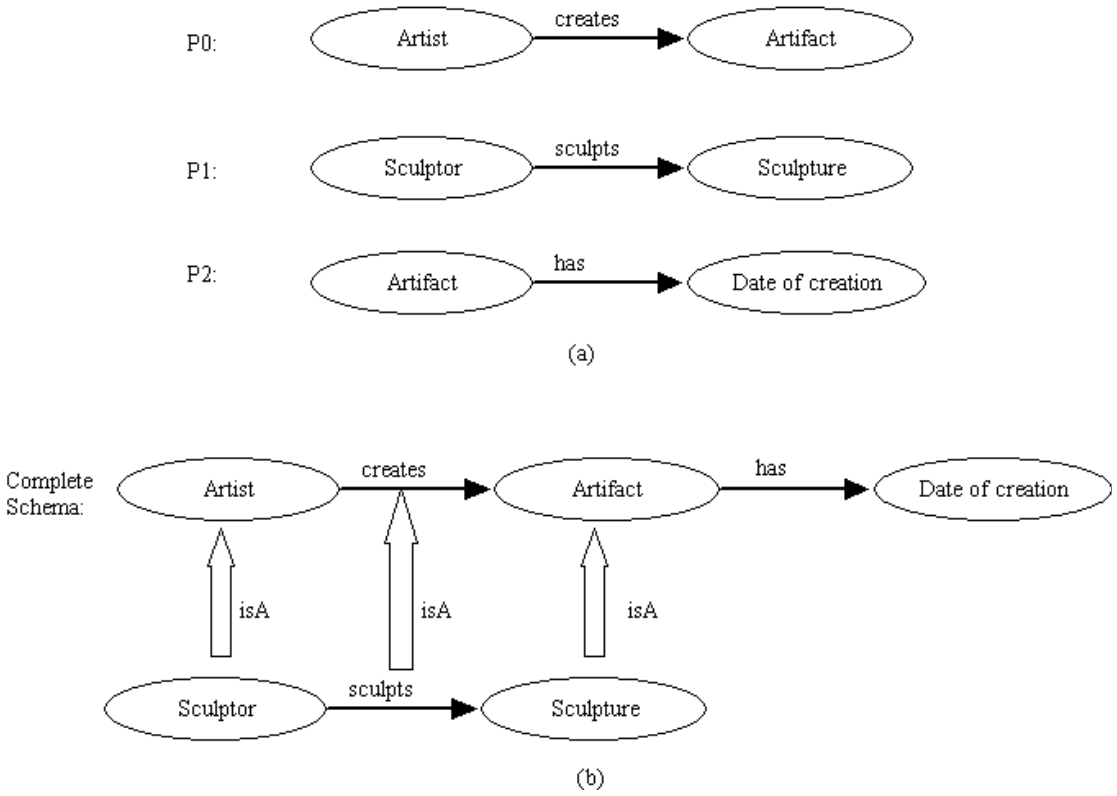
Το πρόβλημα που μας απασχολεί είναι το εξής. Έστω ότι έχουμε ένα δίκτυο P2P στο οποίο οι κόμβοι εκδίδουν τις βάσεις μετά-δεδομένων τους με RDF Σχήματα. *Μπορούμε σε αυτό το δίκτυο να τοποθετούμε τους κόμβους κατά τέτοιο τρόπο, ώστε να κάνουμε πολύπλοκα ερωτήματα*, όπως να ρωτάμε για ένα συγκεκριμένο RDF Σχήμα και η αναζήτηση αυτού του σχήματος (ή των μετά-δεδομένων που εντάσσονται σε αυτό, σαν στιγμιότυπα) να γίνεται *σε αποδοτικό χρόνο*; Σε ένα τέτοιο σύστημα μας ενδιαφέρουν δύο περιπτώσεις ερωτημάτων. Στην πρώτη περίπτωση μας ενδιαφέρει να ρωτάμε για ένα συγκεκριμένο RDF Σχήμα (για ένα γράφο, στην ουσία) και να παίρνουμε σαν απάντηση τα στιγμιότυπα που ανήκουν στις κλάσεις και στις ιδιότητες αυτού του σχήματος. Στην δεύτερη περίπτωση μας ενδιαφέρει να ρωτάμε για τη δομή ενός RDF σχήματος. Δηλαδή, να ρωτάμε για ένα RDF Σχήμα (για ένα γράφο) και σαν απάντηση να παίρνουμε ένα RDF Σχήμα (έναν άλλο γράφο) το οποίο υπάγεται (ο ορισμός της υπαγωγής σχημάτων δίνεται στην υποπαράγραφο 3.3.2) στο RDF Σχήμα του ερωτήματος.

Επίσης, θα θέλαμε και η *εισαγωγή και διαγραφή ενός κόμβου* στο σύστημα να γίνεται σε όσο το δυνατόν μικρότερο χρόνο. Ακόμη, θα επιθυμούσαμε *ο όγκος της πληροφορίας* που κρατάνε οι κόμβοι, προκειμένου να διατηρηθεί η δομή μεταξύ τους,

να είναι όσο το δυνατόν μικρότερος. Για τις τρεις λειτουργίες (εισαγωγή/διαγραφή κόμβου και αναζήτηση ερωτήματος), κατά πρώτο λόγο μας ενδιαφέρει να ελαχιστοποιήσουμε το πλήθος των hops που απαιτούνται και κατά δεύτερο, μας ενδιαφέρει να ελαχιστοποιήσουμε τον αριθμό των μηνυμάτων που πρέπει να ανταλλαχθούν μεταξύ των κόμβων, για να εκτελεστεί μια από αυτές τις λειτουργίες. Πρέπει να τονίσουμε, ότι για να εκμεταλλευτούμε όλες τις δυνατότητες που μας δίνουν τα RDF Σχήματα, θέλουμε η απάντηση που θα πάρουμε να μην προέρχεται, απλά, από μεμονωμένους κόμβους. Με αυτό, εννοούμε ότι δεν αρκεί η απάντηση να προέρχεται μόνο από έναν κόμβο, αλλά αν υπάρχουν πολλά RDF Σχήματα που συγκροτούν το Σχήμα του ερωτήματος, τότε θέλουμε να μας απαντήσουν όλοι οι κόμβοι που εκδίδουν αυτά τα RDF Σχήματα, είτε με τα μετά-δεδομένα τους είτε με τα σχήματά τους (σε αυτή την περίπτωση η απάντηση θέλουμε να έρθει σαν ένα ενιαίο Σχήμα). Περιμένουμε ότι ένα δομημένο σύστημα, θα είναι πολύ πιο αποδοτικό όσον αφορά την αναζήτηση RDF μετά-δεδομένων. Αν όμως χρησιμοποιήσουμε ένα δομημένο σύστημα για να τοποθετούμε τους κόμβους (κατά συνέπεια και τα RDF Σχήματα που φέρουν), τότε θα πρέπει να λύσουμε και το πρόβλημα της συντήρησής του, δηλαδή της εισαγωγής/διαγραφής ενός κόμβου από το σύστημα, ώστε αυτές οι δύο λειτουργίες να γίνονται σε αποδοτικό χρόνο. Από την άλλη, αν χρησιμοποιήσουμε ένα αδόμητο P2P δίκτυο, απαλλασσόμαστε από το πρόβλημα της εισαγωγής/διαγραφής κόμβου, αλλά ο χρόνος απάντησης ενός πολύπλοκου ερωτήματος αυξάνεται δραματικά (σε σχέση με ένα δομημένο σύστημα).

Υπάρχουν διάφορες δομές οι οποίες έχουν μικρό σχετικά κόστος συντήρησης, όπως για παράδειγμα η δομή του Chord [4], η δομή του CAN [5] και η δομή του υπερκύβου [6]. Επίσης, υπάρχουν διάφορες εργασίες, όπως η [7], η [8] και η [9], στις οποίες τα μετά-δεδομένα (ή ολόκληρα σχήματα) αποθηκεύονται σε κάποιο δομημένο σύστημα. Ωστόσο, οι εργασίες αυτές, δεν λαμβάνουν υπόψη ερωτήματα που μπορεί να αναφέρονται στην δομή του συνολικού σχήματος που προκύπτει από τα επιμέρους σχήματα των κόμβων. Για παράδειγμα, έστω ότι το δίκτυο μας περιλαμβάνει τρεις κόμβους, τους P0, P1 και P2 και έχουν ως σχήματα, τους γράφους που φαίνονται στο Σχ. 1a. Σημασιολογικά, ο Γλύπτης (Sculptor) είναι Καλλιτέχνης (Artist) επομένως, σε ένα ενιαίο σχήμα, η κλάση *Sculptor* και η κλάση *Artist* θα συνδέονταν με μια σχέση isA (το ίδιο ισχύει για τις κλάσεις *Sculpture* και *Artifact* των σχημάτων P0 και P1,

αντίστοιχα). Αν κάνουμε μια ερώτηση στα συστήματα [7], [8] και [9], στην οποία να ζητάμε το συνολικό σχήμα του δικτύου, τότε θα μας επιστραφούν τα σχήματα που φαίνονται στο Σχ. 1a, ως ξεχωριστές απαντήσεις. Με αυτές τις απαντήσεις, όμως, δεν φαίνεται η εννοιολογική σύνδεση που έχουν τα 3 σχήματα. Επομένως, θα ήταν πιο λογικό, να μας επιστραφεί μια ολοκληρωμένη απάντηση, όπως αυτή που φαίνεται στο Σχ. 1b, αφού τα 3 σχήματα είναι εννοιολογικά συνδεδεμένα μεταξύ τους. Επίσης, πολλά από τα συστήματα που έχουν προταθεί παρουσιάζουν επιμέρους προβλήματα. Για παράδειγμα, οι εργασίες [7] και [8] στηρίζονται στον υπερκύβο, στον οποίο οι κόμβοι χρειάζεται να «κρατάνε» μεγάλο μέγεθος πληροφορίας, προκειμένου να διατηρηθεί η δομή του υπερκύβου. Επίσης, στην εργασία [9] μπορούμε να διαχειριστούμε μόνο RDF δεδομένα και όχι RDF Σχήματα, κάτι που αποτελεί μεγάλο μειονέκτημα για το συγκεκριμένο σύστημα. Ακόμη, υπάρχουν εργασίες, όπως οι [10] και [11], στις οποίες οι κόμβοι που φέρουν RDF δεδομένα και RDF Σχήματα είναι τοποθετημένοι σε αδόμητο δίκτυο. Σε αυτά, όμως, τα συστήματα ο χρόνος απόκρισης για ένα ερώτημα είναι αρκετά μεγάλος (πολύ πιο μεγάλος απ' ότι στα δομημένα συστήματα) και μεγαλώνει εκθετικά, όσο πιο πολύπλοκο γίνεται το ερώτημα.



Σχήμα 1.1 Παράδειγμα 3 RDF Σχημάτων (a) και η Συνένωσή τους σε Ένα Ενιαίο Σχήμα (b)

Εμείς, θέλουμε ένα δίκτυο, στο οποίο, τόσο η εισαγωγή/διαγραφή ενός κόμβου από το δίκτυο, όσο και η αναζήτηση ενός ερωτήματος (όσο πολύπλοκο κι αν είναι) να γίνεται σε αποδοτικό χρόνο (όσον αφορά τον αριθμό των hops που απαιτούνται), σε σχέση με το μέγεθος του δικτύου. Ακόμη, επιθυμούμε να ελαχιστοποιήσουμε, όσο πιο πολύ γίνεται, τον αριθμό των μηνυμάτων που απαιτούνται για τις λειτουργίες της εισαγωγής/διαγραφής ενός κόμβου από το δίκτυο, της αναζήτησης ενός ερωτήματος. Επίσης, θέλουμε ο χώρος που είναι αναγκασμένοι να κρατάνε οι κόμβοι (για να μπορεί να διατηρείται η δομή), να είναι λογαριθμικός, σε σχέση με το συνολικό αριθμό των κόμβων του δικτύου. Εκτός αυτού, θέλουμε οι κόμβοι που είναι σημασιολογικά ίδιοι, για παράδειγμα, οι κόμβοι με το ίδιο RDF Σχήμα, να δομούνται κάπου κοντά, ώστε να μπορούμε να μεταβαίνουμε από τον ένα κόμβο στον άλλο, σε μικρό χρόνο. Η προσέγγιση που παρουσιάζουμε στην παρούσα εργασία βασίζεται στην ίδια την ιδέα του RDF Σχήματος. Όπως σε ένα RDF Σχήμα, οι κλάσεις και οι ιδιότητες σχηματίζουν μια ιεραρχία, έτσι και στο σύστημά μας, οι κόμβοι δομούνται σημασιολογικά σε ιεραρχία, με βάση το RDF Σχήμα της βάσης μετά-δεδομένων τους.

Ένας κόμβος Α, που έχει Σχήμα SA, το οποίο είναι υποσύνολο του Σχήματος SB ενός κόμβου Β, θα τοποθετηθεί κάτω από τον κόμβο Β ως παιδί του. Αντίστροφα, αν το Σχήμα SA είναι υπερσύνολο του σχήματος SB, τότε ο κόμβος Β θα τοποθετηθεί κάτω από τον Α, ως παιδί του. Έτσι, κόμβοι με ίδιο σχήμα (ή υποσύνολο ενός γενικότερου σχήματος) δομούνται στην ίδια ιεραρχία. Οι ιεραρχίες δομούνται σε ένα δακτύλιο (παρόμοιο με αυτόν του chord) που θα μας βοηθήσει στην αναζήτηση των ερωτημάτων, έτσι ώστε αυτή να γίνεται σε μικρό χρόνο. Ο δακτύλιος αυτός, έχει τα χαρακτηριστικά του δακτυλίου του Chord [4] (*finger tables, successors* κλπ.), κάτι το οποίο μας επιτρέπει να «φθάσουμε» σε μια ιεραρχία σε O(logn) χρόνο, αν n είναι ο συνολικός αριθμός των ιεραρχιών. Με αυτό τον τρόπο, πετυχαίνουμε μια αποδοτική δομή για τη εισαγωγή/ διαγραφή RDF σχημάτων, όσο και για την αναζήτησή τους (αλλά και για το χώρο που απαιτείται να κρατάει κάθε κόμβος).

Το σύστημά μας συγκρίνεται με ένα αδόμητο δίκτυο κόμβων (οι κόμβοι συνδέονται μεταξύ τους τυχαία) και με το σύστημα που προτείνεται στην εργασία [12]. Στην εργασία αυτή, οι κόμβοι δομούνται σε ένα δίκτυο Chord. Τα RDF Σχήματα που εκδίδουν οι κόμβοι, κατακερματίζονται και τοποθετούνται στους κόμβους του chord, ώστε να διατηρείται μια διάταξη. Η διάταξη διατηρείται με βάση μια συγκεκριμένη αρίθμηση, που επιβάλλεται στο RDF Σχήμα που εκδίδει ο κάθε κόμβος. Για την αναζήτηση ενός ερωτήματος σε αυτό το σύστημα, ακολουθείται ένας συγκεκριμένος αλγόριθμος, ο οποίος επισκέπτεται τα RDF Σχήματα που μπορούν να απαντήσουν στο ερώτημα, με τη σειρά διάταξής τους στο Chord. Τα πειράματα μάς έδειξαν ότι το σύστημα μας δουλεύει πολύ καλύτερα (τόσο για την εισαγωγή/διαγραφή κόμβου, όσο και για την αναζήτηση ενός ερωτήματος), όταν ο αριθμός των ιεραρχιών είναι σχετικά μικρός, σε σχέση με το συνολικό αριθμό των κόμβων που υπάρχουν στο δίκτυο. Το γεγονός αυτό αποτελεί πλεονέκτημα για το σύστημά μας, γιατί σε πραγματικά δίκτυα περιμένουμε ότι θα υπάρχει ένας μικρός αριθμός μεγάλων σχημάτων (που ίσως, να περιγράφουν μετά-δεδομένα μεγάλων οργανισμών), τα οποία θα είναι και ριζικά σχήματα στις ιεραρχίες τους, και ένας μεγάλος αριθμός μικρών σχημάτων, τα οποία θα είναι υποσύνολα των πρώτων. Επίσης, το σύστημά μας αποδίδει πολύ καλύτερα σε σχέση με τα άλλα δύο συγκρινόμενα συστήματα, όσον αφορά τον χρόνο εισαγωγής/διαγραφής ενός κόμβου από το σύστημα και την αναζήτηση ενός ερωτήματος. Από την άλλη, το σύστημά μας χρειάζεται να

επικοινωνήσει με πολλούς κόμβους, για να λειτουργήσει η εισαγωγή κόμβου και η αναζήτηση ερωτήματος, με αποτέλεσμα να απαιτεί μεγαλύτερο αριθμό μηνυμάτων από ότι το [12], για αυτές τις δύο λειτουργίες (λιγότερα, όμως, από την περίπτωση του αδόμητου συστήματος). Για να κάνουμε πιο κατανοητή τη χρησιμότητα του συστήματος, που προτείνουμε στην παρούσα διατριβή, ας δούμε το εξής παράδειγμα.

Παράδειγμα 1.1: Ας υποθέσουμε ότι έχουμε ένα σημασιολογικό P2P δίκτυο, στο οποίο οι κόμβοι δημοσιοποιούν τις βάσεις μετά-δεδομένων τους με RDF Σχήματα και ότι τα μετά-δεδομένα τους έχουν σχέση με το θέμα: «*Ζωγραφική*». Ας υποθέσουμε επίσης, ότι θέλουμε να βρούμε όλες τις πηγές του δικτύου, που αναφέρονται σε έργα ενός συγκεκριμένου είδους ζωγραφικής. Για παράδειγμα, πηγές που αναφέρονται στο είδος: «*Ιμπρεσιονισμός*». Αν οι κόμβοι δεν έχουν ομαδοποιηθεί με κάποιο τρόπο, τότε θα έπρεπε να επισκεφθούμε όλους τους κόμβους, για να είμαστε σίγουροι ότι θα πάρουμε όλες τις πηγές που αναφέρονται στο συγκεκριμένο είδος. Στο σύστημά μας ομαδοποιούμε τους κόμβους σε ιεραρχίες, με βάση τα RDF Σχήματά τους. Στην προκειμένη περίπτωση, οι κόμβοι θα μπορούσαν να είχαν ομαδοποιηθεί με βάση το είδος της ζωγραφικής στο οποίο αναφέρονται οι πηγές τους (οι οποίες βρίσκονται στη βάση μετά-δεδομένων κάθε κόμβου). Έτσι, στο σύστημά μας, το ερώτημα θα προωθούνταν, μόνο σε εκείνες τις ομάδες-ιεραρχίες κόμβων, των οποίων το σχήμα περιέχει το είδος του «*Ιμπρεσιονισμού*», εξοικονομώντας χρόνο και απασχολώντας μικρότερο μέρος του δικτύου. Αυτό όμως δεν μας φθάνει. Αν οι ομάδες-ιεραρχίες που δημιουργούνται είναι πολλές (στο παράδειγμά μας, αν υπήρχαν πολλά είδη ζωγραφικής), σε σχέση με τον αριθμό των κόμβων του δικτύου, τότε για να είμαστε σίγουροι ότι θα βρούμε όλες τις πηγές που απαντάνε στο ερώτημά μας, θα πρέπει πάλι να επισκεφθούμε όλες τις ιεραρχίες. Αυτό σημαίνει ότι ο χρόνος που απαιτείται για να πάρουμε την απάντησή μας παραμένει αρκετά μεγάλος. Για να λύσουμε αυτό το πρόβλημα στο σύστημά μας, κάθε ιεραρχία (στην ουσία, ο ριζικός κόμβος κάθε ιεραρχίας) τοποθετείται σε μια δομή δακτυλίου (την οποία θα περιγράψουμε, λεπτομερέστερα, στο Κεφάλαιο 3), ώστε να φθάνουμε σε οποιαδήποτε ιεραρχία σε λογαριθμικό χρόνο, σε σχέση με τον αριθμό των ιεραρχιών.

Συνοπτικά, η παρούσα διατριβή:

- Μελετάει το πρόβλημα της διαχείρισης των RDF Σχημάτων σε δίκτυα P2P. Δηλαδή, μελετάει το πρόβλημα της αναζήτησης ερωτημάτων, που αφορούν RDF Σχήματα, σε ένα P2P δίκτυο, στο οποίο οι κόμβοι εκδίδουν τις βάσεις μετά-δεδομένων τους με RDF Σχήματα. Τα ερωτήματα που μας αφορούν στην παρούσα εργασία χωρίζονται σε δύο κατηγορίες. Στην πρώτη, ένας κόμβος μπορεί να ρωτήσει για ένα RDF Σχήμα και σαν απάντηση να πάρει τα στιγμιότυπα του συγκεκριμένου σχήματος. Στην δεύτερη περίπτωση, ένας κόμβος μπορεί να ρωτήσει, ξανά για ένα RDF Σχήμα και σαν απάντηση να πάρει ένα άλλο RDF Σχήμα, που είναι υποσύνολο του πρώτου. Επίσης, σε ένα τέτοιο P2P δίκτυο, μας ενδιαφέρει οι κόμβοι που είναι σημασιολογικά παρόμοιοι (για παράδειγμα κόμβοι με παρόμοιο RDF Σχήμα), να τοποθετούνται σε κοντινές θέσεις στο δίκτυο. Ακόμη, μας ενδιαφέρει οι κόμβοι να κρατάνε όσο το δυνατόν λιγότερη πληροφορία (πληροφορία που είναι απαραίτητη για να διατηρηθεί δομή μεταξύ των κόμβων). Επίσης, μας ενδιαφέρει η εισαγωγή/διαγραφή κόμβου στο σύστημα, να γίνεται όσο το δυνατόν πιο γρήγορα (όσον αφορά τον αριθμό των hops) και να απαιτεί όσο το δυνατόν πιο λίγα μηνύματα.
- Παρουσιάζει ένα δομημένο P2P σύστημα για την αποδοτική λύση του προβλήματος. Στο σύστημα αυτό, οι κόμβοι δομούνται σε μια ιεραρχία και ο ριζικός κόμβος κάθε ιεραρχίας δομείται σε μια δομή δακτυλίου, ώστε να μπορούμε να μεταβαίνουμε από μια ιεραρχία σε μια άλλη σε σύντομο χρόνο.
- Συγκρίνει πειραματικά το σύστημα που προτείνουμε με άλλα συστήματα της υπάρχουσας βιβλιογραφίας

Το υπόλοιπο της εργασίας είναι δομημένο ως εξής. Στο επόμενο Κεφάλαιο, παρουσιάζονται τα στοιχεία, τα οποία χρησιμοποιούμε σαν δεδομένα στο σύστημά μας, δηλαδή μια περιγραφή των RDF δεδομένων και του RDF Σχήματος. Επίσης παρουσιάζεται μια περιγραφή των P2P συστημάτων, καθώς και το πρόβλημα που προσπαθούμε να επιλύσουμε στην παρούσα διατριβή. Στο Κεφάλαιο 3, περιγράφουμε την τοπολογία του συστήματός μας και τις διαδικασίες που γίνονται κατά την εισαγωγή/διαγραφή ενός κόμβου στο σύστημα και κατά την αναζήτηση ενός ερωτήματος. Επίσης, στο ίδιο Κεφάλαιο παρουσιάζεται μια τεχνική εξισορρόπησης φόρτου, όταν το σύστημά μας υπερφορτώνεται. Στο Κεφάλαιο 4, περιγράφουμε τον

τρόπο υλοποίησης του συστήματός μας, καθώς και των δύο συστημάτων με τα οποία συγκρίναμε το δικό μας σύστημα. Στο Κεφάλαιο 5, παρουσιάζονται τα αποτελέσματα των πειραμάτων, ενώ στο Κεφάλαιο 6, παρουσιάζουμε τη σχετική βιβλιογραφία. Τέλος, στο Κεφάλαιο 7 συζητάμε για τη μελλοντική δουλεία που μπορεί να γίνει πάνω στο σύστημά μας και παρουσιάζουμε, συνοπτικά, τα συμπεράσματα της παρούσας εργασίας.

ΚΕΦΑΛΑΙΟ 2. ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ

2.1. Συστήματα Ομοτίμων

Τα συστήματα ομοτίμων (peer-to-peer) είναι κατανεμημένα δίκτυα κόμβων (peers), στα οποία οι κόμβοι συνδέονται μεταξύ τους με διάφορους τρόπους και τεχνικές. Η σύνδεση αυτή γίνεται πάνω από το φυσικό TCP/IP δίκτυο, χωρίς να απαιτείται κάποιος κεντρικός έλεγχος.

Πριν από τα P2P συστήματα επικρατούσαν τα λεγόμενα *client-server* (ή *κεντρικοποιημένα*) συστήματα, στα οποία πολλοί απλοί κόμβοι (*clients*) συνδέονταν με έναν πιο ισχυρό κόμβο (*server*). Ο δεύτερος αναλάμβανε όλες τις διαδικασίες για εισαγωγή/διαγραφή κόμβου από το σύστημα, καθώς και τις προωθήσεις των ερωτημάτων. Το μεγάλο μειονέκτημα, όμως, αυτών των δικτύων είναι ότι δεν είναι ανεκτικά σε σφάλματα. Δηλαδή, όταν ένας κόμβος-server αποτύχει (πράγμα, που μπορεί να γίνει συχνά, αφού ο server δέχεται υψηλό όγκο πληροφορίας και δεν μπορεί να ανταπεξέλθει πάντα), τότε ολόκληρο το δίκτυο δεν μπορεί να λειτουργήσει. Ένα ακόμη μειονέκτημα αυτών των δικτύων αποτελεί το γεγονός, ότι δεν είναι

κλιμακούμενα. Δηλαδή, για μεγάλο αριθμό κόμβων, είτε υπολειτουργούν (η αναζήτηση ερωτημάτων διαρκεί πάρα πολύ) είτε δεν λειτουργούν καθόλου. Τα μειονεκτήματα αυτά ήρθαν να διορθώσουν τα P2P συστήματα.

Το χαρακτηριστικό των P2P συστημάτων είναι ότι όλοι οι κόμβοι είναι ομότιμοι, δηλαδή έχουν τις ίδιες δυνατότητες, κρατούν ίδιο μέγεθος πληροφορίας και έχουν τις ίδιες ευθύνες. Το γεγονός αυτό κάνει τα συστήματα P2P κλιμακούμενα, αφού απ' όλους τους κόμβους περνάει ο ίδιος όγκος πληροφορίας (για να το εξασφαλίσουν αυτό, όσο είναι δυνατόν, πολλά συστήματα P2P που έχουν προταθεί, ενσωματώνουν και τεχνικές εξισορρόπησης φόρτου). Επίσης, τα διάφορα συστήματα P2P, που υπάρχουν στη βιβλιογραφία, εγγυώνται ανεκτικότητα σε σφάλματα. Δηλαδή, αν χαλάσει ένας κόμβος, τότε τα συστήματα αυτά, παρέχουν τεχνικές, ώστε το δίκτυο να λειτουργεί κανονικά. Ακόμη, τα P2P συστήματα εγγυώνται πολύ γρήγορους χρόνους, όσον αφορά την εισαγωγή/διαγραφή κόμβων στα συστήματα και την αναζήτηση ερωτημάτων (συνήθως οι χρόνοι αυτοί είναι λογαριθμικοί, σε σχέση με το αριθμό των κόμβων).

Στα αρχικά συστήματα ομοτίμων κόμβων (Napster) υπήρχε ένας κεντρικός κόμβος ο οποίος κράταγε περισσότερη πληροφορία από τους άλλους και ήταν υπεύθυνος για την αναζήτηση αντικειμένων, καθώς και για την εισαγωγή ή διαγραφή ενός κόμβου. Αργότερα αναπτύχθηκαν τα πλήρως κατανεμημένα δίκτυα P2P (Gnutella), τα οποία όμως, για μεγάλο αριθμό κόμβων παρουσίαζαν προβλήματα (μεγάλος χρόνος στην αναζήτηση ερωτημάτων). Σήμερα, παρουσιάζονται νέα συστήματα ομοτίμων κόμβων που προσπαθούν να λύσουν τέτοια προβλήματα. Αυτό που μας ενδιαφέρει στα P2P συστήματα (κι αυτό που θα μελετηθεί) είναι να γίνονται, όσο το δυνατόν πιο αποδοτικά, οι λειτουργίες εισαγωγής/διαγραφής ενός κόμβου (ή δεδομένων) και αναζήτησης ενός ερωτήματος. Όσον αφορά την αναζήτηση ερωτήματος, μας ενδιαφέρει να βρούμε τρόπους, οι οποίοι θα μας δίνουν αποτέλεσμα σε μικρό χρόνο. Η μετρική σε αυτή την περίπτωση είναι τα *hops*, δηλαδή ο αριθμός των βημάτων που απαιτείται, μεταξύ των κόμβων, για να βρεθεί η απάντηση του ερωτήματος. Επίσης, μας ενδιαφέρει η αναζήτηση να γίνεται με ανταλλαγή, όσο το δυνατόν, λιγότερων μηνυμάτων. Η μετρική σε αυτή την περίπτωση είναι ο συνολικός αριθμός των μηνυμάτων. Για την εισαγωγή και διαγραφή κόμβων, μας ενδιαφέρει να βρούμε

τρόπους ώστε η εισαγωγή ή διαγραφή ενός κόμβου να γίνεται γρήγορα και να επηρεάζει όσο το δυνατόν λιγότερους κόμβους στο δίκτυο. Επίσης, σε μερικές περιπτώσεις μας ενδιαφέρει και η αντιγραφή δεδομένων σε πολλούς κόμβους έτσι ώστε να γίνονται πιο προσιτά.

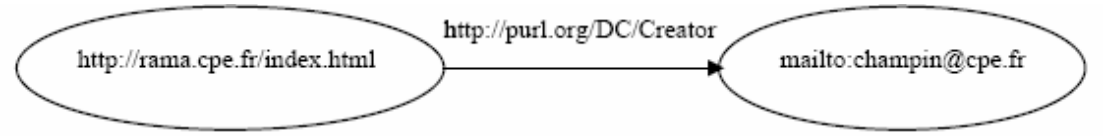
Ανάλογα με τον τρόπο με τον οποίο συνδέονται οι κόμβοι μεταξύ τους, τα συστήματα P2P χωρίζονται σε δομημένα και μη δομημένα συστήματα. Στα δομημένα, οι κόμβοι συνδέονται με κάποιο συγκεκριμένο τρόπο, έτσι ώστε να δημιουργήσουν μια δομή. Χαρακτηριστικά συστήματα σε αυτή την κατηγορία είναι το Chord και το CAN. Τα δύο αυτά συστήματα χρησιμοποιούν κατανεμημένους πίνακες κατακερματισμού και ζεύγη κλειδιών - τιμών για να αποθηκεύσουν ή να ανακτήσουν τα δεδομένα τους. Στα μη δομημένα συστήματα, οι κόμβοι συνδέονται τυχαία μεταξύ τους. Τα συστήματα αυτά πλεονεκτούν, σε σχέση με τα δομημένα συστήματα, στο ότι δεν απαιτείται από τους κόμβους που συμμετέχουν σε αυτό, να μεριμνήσουν για τη συντήρηση του δικτύου (δηλαδή, να αποθηκεύσουν κάποια δεδομένα που θα βοηθήσουν στην διατήρηση του δικτύου, κατά την εισαγωγή ή διαγραφή ενός κόμβου από το δίκτυο). Από την άλλη, τα αδόμητα συστήματα μειονεκτούν σε σχέση με τα P2P, στο ότι η αναζήτηση των δεδομένων δεν γίνεται αρκετά γρήγορα. Για να λύσουν το πρόβλημα της αναζήτησης ερωτημάτων, τα συστήματα αυτά χρησιμοποιούν διάφορους αλγόριθμους, όπως ο αλγόριθμος *επαναληπτικής εκβάθυνσης* [13] και ο αλγόριθμος *Random Walks* [13].

2.2. Περιγραφή Δεδομένων

Τα δεδομένα που χρησιμοποιούμαι στο σύστημά μας είναι RDF δεδομένα και RDF Σχήματα. Τα RDF Σχήματα θα τα χρησιμοποιήσουμε για να περιγράψουμε τα μετά-δεδομένα μας και για να ταξινομήσουμε τους κόμβους σε διαφορετικές ιεραρχίες.

Το RDF (*Resource Description Framework*) είναι ένα πλαίσιο που μας επιτρέπει να περιγράψουμε πηγές δεδομένων στο διαδίκτυο, όπως ένα web site και το περιεχόμενό του. Οι RDF περιγραφές (ονομάζονται και μετά-δεδομένα) μπορούν να περιλαμβάνουν τους συγγραφείς της πηγής, την ημερομηνία δημιουργίας της πηγής,

κλειδιά για μηχανές αναζήτησης, κλπ. Τα RDF δεδομένα μπορούν να περιγραφούν στο πλαίσιο της XML [14]. Το βασικό δομικό στοιχείο των RDF δεδομένων [15] είναι η *τριάδα* (*statement*), η οποία αποτελείται από το *υποκείμενο* (*subject*), το *αντικείμενο* (*object*) και το *κατηγόρημα* (*predicate*). Πιο συγκεκριμένα, το υποκείμενο είναι μια πηγή, η οποία αφορά την τοποθεσία στην οποία έγινε η RDF δήλωση, το κατηγόρημα είναι μια ιδιότητα του υποκειμένου και το κατηγορούμενο είναι η τιμή αυτής της ιδιότητας. Το υποκείμενο και το κατηγόρημα είναι κάποιες πηγές δεδομένων και καθορίζονται, από ένα URI, μια διεύθυνση δηλαδή του διαδικτύου. Το αντικείμενο μπορεί να είναι ένα URI ή ένα απλό string. Μια τριάδα RDF μπορεί να παρασταθεί σαν ένας κατευθυνόμενος γράφος, όπου το υποκείμενο ενώνεται με το αντικείμενο με μια ακμή, από το υποκείμενο προς το αντικείμενο, η οποία ονοματίζεται με το κατηγόρημα. Ένα παράδειγμα RDF τριάδας φαίνεται στο Σχ. 2.1 και η αντίστοιχη XML περιγραφή του στο Σχ. 2.2. Στο παράδειγμα αυτό, ο champin είναι ο δημιουργός του index.html. Παρατηρούμε πως και το υποκείμενο και το αντικείμενο, αλλά και το κατηγόρημα εκφράζονται ως URI διευθύνσεις. Επίσης, θα πρέπει να τονίσουμε ότι ένα υποκείμενο μπορεί να έχει πολλές τιμές για ένα συγκεκριμένο κατηγόρημα. Συνδεόμενες τριάδες (δηλαδή το αντικείμενο μιας τριάδας είναι υποκείμενο μιας άλλης τριάδας ή και το αντίστροφο, κ.ο.κ.) σχηματίζουν ένα κατευθυνόμενο άκυκλο γράφο.



Σχήμα 2.1 Παράδειγμα RDF Τριάδας

Το RDF Σχήμα [1] είναι μια γλώσσα περιγραφής ενός συνόλου RDF δεδομένων. Αποτελεί μια σημασιολογική επέκταση των RDF δεδομένων. Παρέχει μηχανισμούς για την περιγραφή πηγών δεδομένων καθώς και για τις σχέσεις μεταξύ αυτών των πηγών. Ένα RDF Σχήμα μπορεί επίσης να περιγραφεί με τη γλώσσα XML και με μερικά καινούρια χαρακτηριστικά που περιγράφονται αναλυτικά στο [1]. Στο Σχ. 2.3, φαίνεται μια περιγραφή ενός RDF Σχήματος σε XML-RDF. Η σχηματική περιγραφή φαίνεται στο Σχ. 2.4. Το συγκεκριμένο RDF Σχήμα δηλώνει ότι ο Lucas είναι άντρας

(*Man*) και έχει μητέρα (*hasMother*) τη Laura. Το RDF Σχήμα αποτελείται από ένα σύνολο τριάδων (*schema statements*). Τα statements αυτά περιλαμβάνουν δύο *κλάσεις* (*classes*) οι οποίες ενώνονται από μία *ιδιότητα* (*property*) σχηματίζοντας ένα κατευθυνόμενο γράφο. Η κλάση από την οποία ξεκινάει μια ιδιότητα λέγεται *πεδίο ορισμού* της ιδιότητας (*domain*), ενώ η κλάση στην οποία καταλήγει μια ιδιότητα λέγεται *πεδίο τιμών* (*range*) της ιδιότητας. Συνήθως, ένα σύνολο ομοειδών υποκείμενων και το κατηγόρημα εκφράζονται σαν κλάσεις (class) αντικειμένων (δηλαδή, σαν ένα σύνολο αντικειμένων το οποίο περιλαμβάνει κάποια URIs), ενώ το κατηγόρημα εκφράζεται σαν ιδιότητα (property) μεταξύ κλάσεων. Επίσης, μεταξύ των κλάσεων, αλλά και σχημάτων, ορίζονται σχέσεις *υπαγωγής* (*subsumption*). Θα λέμε ότι μια κλάση ή μια ιδιότητα υπάγεται σε μια άλλη κλάση ή ιδιότητα, αντίστοιχα, όταν όλα τα αντικείμενα της πρώτης ανήκουν και στη δεύτερη (οι δύο κλάσεις συνδέονται με μια σχέση isA). Η ίδια ιδιότητα ισχύει και για τα σχήματα. Δηλαδή, θα λέμε ότι ένα σχήμα υπάγεται σε ένα άλλο, όταν όλα τα statements του πρώτου, ανήκουν ή υπάγονται στο δεύτερο. Η υπαγωγή ενός σχήματος μπορεί να είναι *οριζόντια* (*horizontal*) ή *κάθετη* (*vertical*). Πιο συγκεκριμένα, στους ορισμούς 2.1 και 2.2, ορίζονται και τυπικά οι σχέσεις υπαγωγής μεταξύ δύο Σχημάτων S_1 και S_2.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<rdf: RDF xmlns:rdf = "http://www.w3org/1999/02/22 rdf-syntax-ns#"
          xmlns:dc = "http://www.purl.org/DC"
   <rdf:Description about = "http://rama.cpe.fr/ index.html">
     <dc:Creator>
       <rdf:Description about = "mailto@champin@cpe.fr">
     </dc:Creator>
   </rdf:Description>
</rdf:RDF>
```

Σχήμα 2.2 XML περιγραφή RDF τριάδας

Ορισμός 2.1: Έστω δύο RDF Σχήματα S_1 και S_2. Θα λέμε ότι το S_1 υπάγει οριζόντια το S_2 ή ότι το S_2 υπάγεται οριζόντια από το S_1, αν και μόνο αν κάθε statement (τριάδα) του S_2 ανήκει και στο S_1.

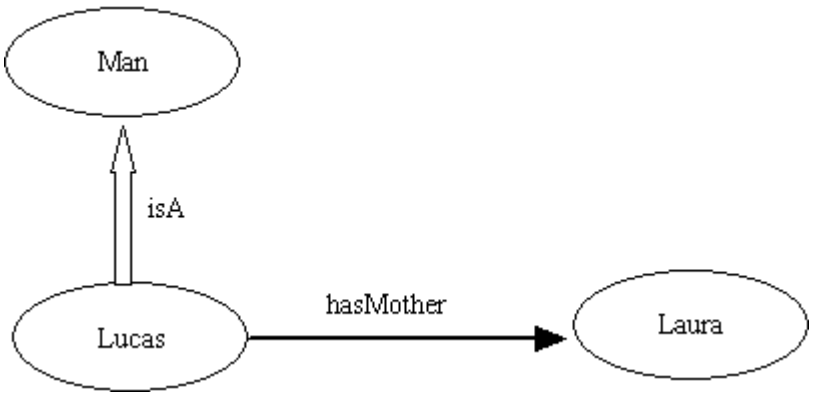
```
<rdf:RDF xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://www.inria.fr/2006/12/05/humans.rdfs#"
  xml:base="http://www.inria.fr/2006/12/05/humans.rdfs-instances" >

<rdf:Description rdf:ID="Lucas">
 <rdfs:type rdf:resource="http://www.inria.fr/2006/12/05/humans.rdfs#Man"/>
 <hasMother rdf:resource="#Laura"/>
</rdf:Description>
```

Σχήμα 2.3 Περιγραφή RDF Σχήματος σε XML-RDF

Ορισμός 2.2: Έστω δύο RDF Σχήματα S_1 και S_2. Θα λέμε ότι το S_1 υπάγει κάθετα το S_2 ή ότι το S_2 υπάγεται κάθετα από το S_1, αν και μόνο αν για κάθε statement (τριάδα) st_2 του S_2, υπάρχει ένα statement (τριάδα) st_1 του S_1 ώστε:

i. για κάθε κλάση C του st_2 να υπάρχει μια κλάση C' του st_1, ώστε να ισχύει C isA C' ή C=C' και
ii. για την ιδιότητα P του st_2 και την ιδιότητα P' του st_1, ισχύει P isA P' ή P=P'.

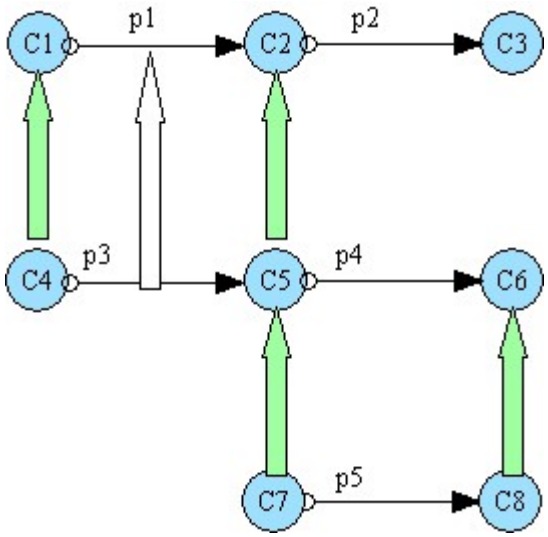
Ορισμός 2.3: Έστω δύο RDF Σχήματα S_1 και S_2. Θα λέμε ότι το S_1 υπάγει το S_2 ή ότι το S_2 υπάγεται από το S_1 (εναλλακτικά μπορούμε να πούμε ότι το S_1 είναι υπερσχήμα του S_2 ή ότι το S_2 είναι υποσχήμα του S_1), αν και μόνο αν το S_1 υπάγει οριζόντια ή (και) κάθετα το S_2



Σχήμα 2.4 RDF Σχήμα

Στο Κεφάλαιο 3, παρουσιάζουμε έναν πιο φορμαλιστικό ορισμό (*Ορισμός 3.2*) για την υπαγωγή δύο Σχημάτων.

Για να παρουσιάσουμε τις σχέσεις υπαγωγής με ένα παράδειγμα, ας δούμε το Σχ. 2.5. Έστω, λοιπόν, το RDF Σχήμα που φαίνεται στο Σχ. 2.5, το οποίο παρέχεται από έναν κόμβο N1. Παρατηρούμε ότι οι κλάσεις C4 και C5 υπάγονται στις κλάσεις C1 και C2 αντίστοιχα. Η ιδιότητα ρ1 υπάγει την ιδιότητα ρ3, ενώ και οι κλάσεις C5 και C6 υπάγουν τις κλάσεις C7 και C8 αντίστοιχα. Έστω ένας κόμβος N2 με RDF σχήμα C1→C2. Τότε, λέμε ότι το σχήμα του N2 υπάγεται οριζόντια στο σχήμα του N1. Έστω, επίσης, ένας κόμβος N3 με σχήμα C7→C8. Τότε, λέμε ότι ο το σχήμα του κόμβου N3 υπάγεται κάθετα στο σχήμα του κόμβου N1. Επίσης, μπορούμε να παρατηρήσουμε ότι μπορούν να δημιουργηθούν ιεραρχίες από RDF σχήματα και αντίστοιχα από κόμβους, που παρέχουν αυτά τα σχήματα. Κάτι τέτοιο θα γινόταν, αν είχαμε έναν κόμβο N4 με σχήμα C7, οπότε αυτό θα υπαγόταν στο σχήμα του N3, σχηματίζοντας την ιεραρχία N2 ← N1 → N3 → N4.



Σχήμα 2.5 Παράδειγμα RDF Σχήματος. Με Κύκλους Απεικονίζονται οι Κλάσεις, Με Απλά Βέλη οι Ιδιότητες, Με Σκούρα οι Σχέσεις Υπαγωγής (Οριζόντιας) Κλάσεων, ενώ με Λευκά Βέλη οι Σχέσεις Υπαγωγής Ιδιοτήτων

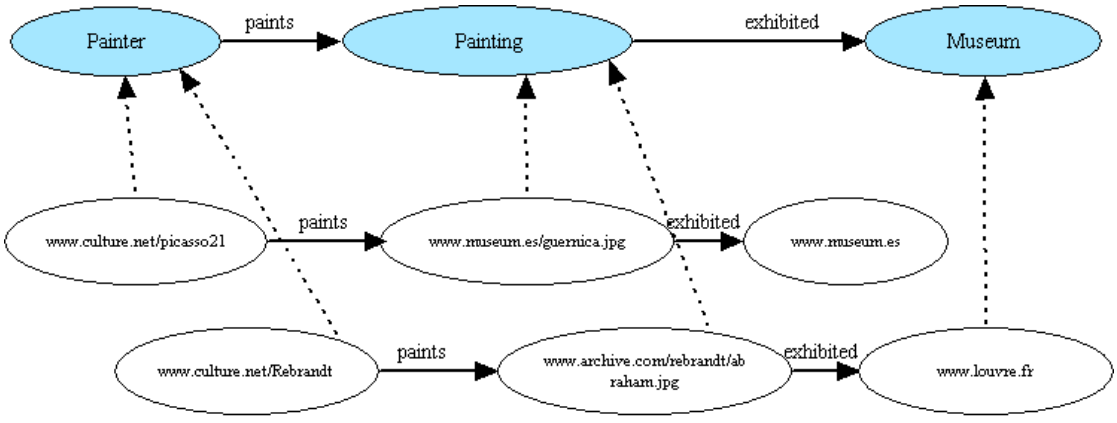
2.3. Περιγραφή Ερωτημάτων

Στο σύστημά μας, μας απασχολούν δύο ομάδες ερωτημάτων. Στην πρώτη ομάδα ερωτημάτων κάθε κόμβος μπορεί να ρωτήσει για ένα RDF Σχήμα, δηλαδή, για έναν γράφο σε επίπεδο κλάσεων και ιδιοτήτων. Η ερώτηση περιλαμβάνει ως είσοδο ένα

RDF σχήμα υπό αναζήτηση. Η απάντηση περιλαμβάνει έναν γράφο (ή γράφους) από τα στιγμιότυπα του συγκεκριμένου RDF Σχήματος, ήτοι, RDF γράφους των οποίων οι κόμβοι συνδέονται με ακμές τύπου *στιγμιοτύπου* με τους κόμβους του γράφου ερώτησης.

Παράδειγμα 2.1

Για παράδειγμα, στο Σχ. 2.6 βλέπουμε ένα RDF Σχήμα και κάποια από τα στιγμιότυπά του. Πιο συγκεκριμένα, φαίνεται το RDF Σχήμα *Painter --- paints ---> Painting --- exhibited ---> Museum*. Η κλάση *Painter* περιλαμβάνει τα στιγμιότυπα *www.culture.net/picasso21* και *www.culture.net/Rebrandt*. Η κλάση *Painting* περιλαμβάνει τα στιγμιότυπα *www.meuseum.es/guernica.jpg* και *www.archive.com/rebrandt/abraham.jpg*. Η κλάση *Meuseum* περιλαμβάνει τα στιγμιότυπα *www.meuseum.es* και *www.louvre.fr*. Τα κατηγορήματα που συνδέουν τα RDF δεδομένα φαίνονται στο Σχ. 2.6. Έστω ότι ένας κόμβος ρωτάει για «όλους τους καλλιτέχνες και τα δημιουργήματά τους που εκτίθενται σε κάποιο μουσείο». Το σχήμα του ερωτήματος αυτού φαίνεται στο Σχ. 2.6 με τους κόμβους σε έγχρωμο φόντο. Οι απαντήσεις που θα πάρει ο κόμβος θα είναι 2 γράφοι με τα στιγμιότυπα του συγκεκριμένου σχήματος (σε άσπρο φόντο στο Σχ. 2.6) και πιο συγκεκριμένα, οι γράφοι *www.culture.net/picasso21--- paints ---> www.meuseum.es/guernica.jpg --- exhibited ---> www.meuseum.es* και *www.culture.net/Rebrandt --- paints ---> www.archive.com/rebrandt/abraham.jpg --- exhibited ---> www.louvre.fr*.

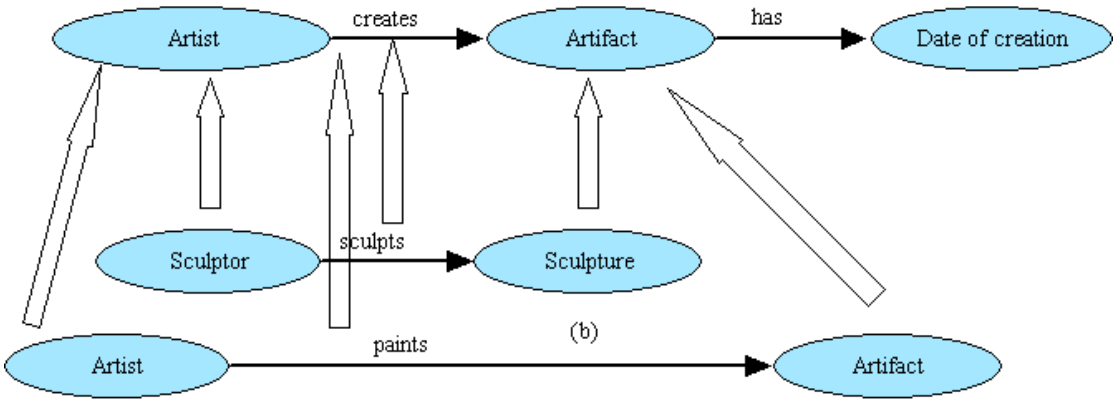


Σχήμα 2.6 RDF Σχήμα και Στιγμιότυπά του. Με Σκούρο Χρώμα Φαίνεται το RDF Σχήμα και με Λευκό Χρώμα Φαίνονται τα RDF Δεδομένα

Στη δεύτερη ομάδα ερωτημάτων, ένας κόμβος μπορεί να ρωτήσει για τη δομή ενός RDF Σχήματος (ένας γράφος σε επίπεδο κλάσεων και ιδιοτήτων). Ξανά, η ερώτηση περιλαμβάνει ένα γράφο RDF Σχήματος. Οι απαντήσεις που θα πάρει είναι κάποιο ή κάποια RDF Σχήματα (ένας ή περισσότεροι γράφοι σε επίπεδο κλάσεων και ιδιοτήτων) που υπάγονται στο RDF Σχήμα του ερωτήματος.

Παράδειγμα 2.2

Για παράδειγμα, στο Σχ. 2.7, βλέπουμε ένα καθολικό RDF Σχήμα. Έστω ότι ένας κόμβος ρωτάει για το πιο γενικό σχήμα που υπάγεται κάθετα στο σχήμα *Artist---creates--->Artifact*. Η απάντηση σε αυτό το ερώτημα θα είναι το RDF Σχήμα (δηλαδή, ένας γράφος από κλάσεις και ιδιότητες) που φαίνεται στο Σχ. 2.8.

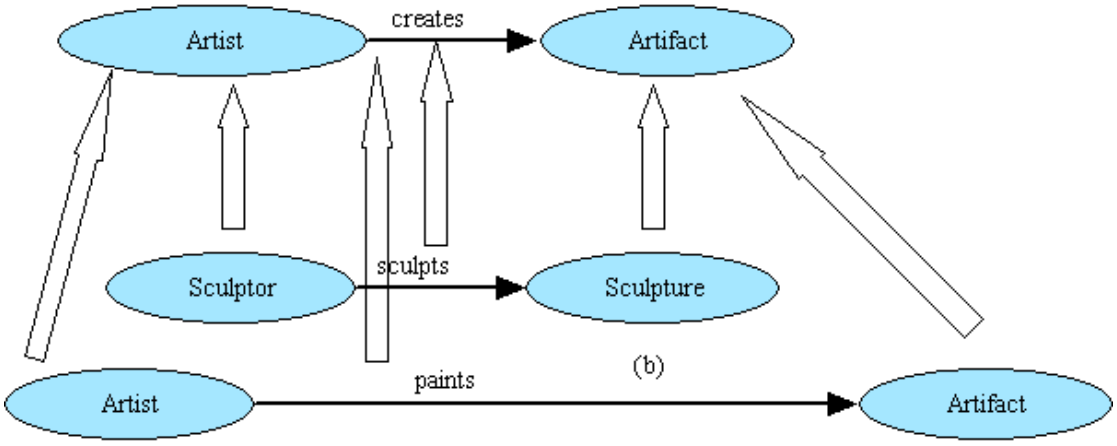


Σχήμα 2.7 Καθολικό RDF Σχήμα

2.4. Περιγραφή του Προβλήματος

Τα περισσότερα P2P δίκτυα που υπάρχουν σήμερα ([2], [3] κλπ.) είναι κατάλληλα για απλές λειτουργίες, όπως ανταλλαγή μουσικών αρχείων. Ωστόσο, η ανάπτυξη των μετά-δεδομένων ως μέσο για την περιγραφή πηγών δεδομένων (που βρίσκονται σε κάποιους κόμβους του δικτύου), έφερε την ανάγκη για την δημιουργία συστημάτων για την αποδοτική αποθήκευση και αναζήτηση τέτοιων μετά-δεδομένων σε P2P δίκτυα. Εκτός, όμως, από την απλή διαχείριση των μεταδεδομένων, θα επιθυμούσαμε να εκμεταλλευτούμε στο έπακρο τους τα σημασιολογικά δίκτυα που μπορούν να προκύψουν από κόμβους οι οποίοι δημοσιοποιούν τις βάσεις δεδομένων τους (ή

μετά-δεδομένων τους) με σχήματα, όπως τα RDF Σχήματα. Σε αυτά τα δίκτυα μπορούμε να ρωτήσουμε πιο σύνθετες ερωτήσεις που έχουν σχέση, ακριβώς με τη σημασιολογία που εκφράζει τη δεδομένη στιγμή το δίκτυο ή ένα μέρος του δικτύου (τέτοια παραδείγματα ερωτημάτων θα δούμε στην Παράγραφο 3.2), κάτι που δεν μπορούμε να κάνουμε σε δίκτυα όπως το Chord ή το CAN.



Σχήμα 2.8 RDF Σχήμα

Προς αυτή τη κατεύθυνση έχουν περιγραφεί συστήματα που στηρίζονται στα αδόμητα δίκτυα, όπως τα [10], [11] και στα δομημένα συστήματα, όπως τα [7], [8] (δομή υπερκύβου), [12] (δομή Chord) και [16] (δομή P-Grid [17]). Τα συστήματα [10] και [11] παρουσιάζουν μεγάλο χρόνο αναζήτησης ερωτημάτων, ενώ τα υπόλοιπα, παρουσιάζουν διαφορετικά προβλήματα ή επικεντρώνουν σε ένα μικρό μέρος των δυνατοτήτων των σημασιολογικών δικτύων (που βασίζονται σε RDF Σχήματα). Για παράδειγμα, τα συστήματα [7] και [8], που βασίζονται σε υπερκύβους, απαιτούν από τους κόμβους να κρατάνε δεδομένα για την επόμενη δομή υπερκύβου (όπως περιγράφεται στο [6]), πράγμα που αυξάνει δραματικά το χώρο που πρέπει να κρατάνε οι κόμβοι. Επίσης, τα συστήματα αυτά, δεν εκμεταλλεύονται τη σημασιολογία που έχει κάθε φορά το δίκτυο. Για παράδειγμα, σε ένα ερώτημα που αφορά το σχήμα ολόκληρου του δικτύου, τα συστήματα αυτά, θα απαντούσαν με τα μεμονωμένα σχήματα του κάθε κόμβου και όχι με ένα ενιαίο σχήμα. Τα συστήματα που προτείνονται στα [12] και [16] μπορούν να απαντήσουν σε ένα ερώτημα που αφορά τη δομή του συνολικού σχήματος που υπάρχει σε ένα δίκτυο κάθε φορά (άρα

μπορούν να εκμεταλλευτούν τη σημασιολογία του δικτύου), αλλά επικεντρώνονται μόνο στο χρόνο απόκρισης ερωτημάτων, χωρίς να τους ενδιαφέρουν, τόσο, οι χρόνοι εισαγωγής και διαγραφής ενός κόμβου από το σύστημα (κάτι που είναι αρκετά σημαντικό στα P2P δίκτυα).

Εμείς, αναζητούμε μια δομή που θα εκμεταλλεύεται πλήρως τη δομή του σχήματος ενός μέρους του δικτύου, αλλά και τη δομή σχημάτων επιμέρους μερών του συνολικού δικτύου. Επίσης, θα θέλαμε το σύστημά μας να έχει αποδοτικούς χρόνους (με βάση τα συνολικά hops που απαιτούνται), όσον αφορά την αναζήτηση ενός ερωτήματος και την εισαγωγή/διαγραφή ενός κόμβου. Για να το πετύχουμε αυτό, θα δημιουργήσουμε ιεραρχίες κόμβων. Κάθε ιεραρχία θα περιέχει κόμβους, των οποίων τα σχήματα, θα είναι υποσύνολα (ή θα υπάγονται, όπως εξηγούμε στην Παράγραφο 2.3) των κόμβων-προγόνων τους, ενώ τα σχήματα όλων των κόμβων μιας ιεραρχίας θα είναι υποσύνολα του σχήματος του ριζικού κόμβου. Για να συνδέσουμε αυτές τις ιεραρχίες, θα χρησιμοποιήσουμε ένα δακτύλιο σαν αυτό του chord, προκειμένου να μπορούμε να εντοπίσουμε μια συγκεκριμένη ιεραρχία γρήγορα και να πετύχουμε ικανοποιητικούς χρόνους, όσον αφορά την εισαγωγή/διαγραφή κόμβων.

2.5. Υποθέσεις για το Σύστημά μας

Σε αυτή την παράγραφο παρουσιάζουμε όλες τις υποθέσεις που κάνουμε, για το σύστημα που παρουσιάζουμε στην παρούσα διατριβή. Στη συνέχεια της διατριβής ακολουθούμε αυτές τις υποθέσεις, χωρίς κάθε φορά να τονίζουμε την κάθε υπόθεση.

2.5.1. Υποθέσεις για τα Σχήματα των Κόμβων

Από εδώ και στο εξής, θα αναφερόμαστε σε σχήματα κόμβων, χωρίς κάθε φορά να αναφέρουμε ότι πρόκειται για RDF Σχήματα. Κάθε RDF Σχήμα αποτελείται από ένα σύνολο statements, όπως είπαμε και στην παράγραφο 2.2. Εναλλακτικά, θα αναφερόμαστε στα statements ενός σχήματος ως τριάδες. Επίσης, το ριζικό σχήμα μιας ιεραρχίας σχημάτων θα αναφέρεται εναλλακτικά ως υπερσχήμα.

Στο σύστημά μας, κάθε κόμβος που θέλει να ενταχθεί στο σύστημα, θα πρέπει να εκδώσει το σχήμα της βάσης μετά-δεδομένων του. Τα σχήματα που μπορεί να εκδώσει ένας κόμβος δεν είναι αυθαίρετα, αλλά υπακούν σε ένα καθολικό RDF Σχήμα. Τα σχήματα, θα πρέπει να υπάγονται στο καθολικό αυτό RDF Σχήμα. Η ύπαρξη του καθολικού σχήματος, μας επιτρέπει να λύσουμε προβλήματα ονοματολογίας των κλάσεων και των ιδιοτήτων, που μπορεί να υπήρχαν μεταξύ των διαφόρων σχημάτων, εάν αυτά δεν συμφωνούσαν με την ονοματολογία ενός καθολικού σχήματος. Επίσης, θα μας βοηθήσει στο να «κωδικοποιήσουμε» τα σχήματα με τον παρακάτω τρόπο.

Ουσιαστικά, το καθολικό σχήμα είναι μια ιεραρχία από κόμβους-κλάσεις (μερικές από τις κλάσεις συνδέονται μεταξύ τους με properties (εναλλακτικά, τις properties θα τις ονομάζουμε ιδιότητες). Σε κάθε κόμβο-κλάση αυτής της ιεραρχίας, θέτουμε ένα αριθμό- ταυτότητα (id), αριθμώντας την ιεραρχία με pre-order αρίθμηση. Η αρίθμηση αυτή, έχει την εξής ιδιότητα. Κόμβοι της ιεραρχίας, που είναι απόγονοι κάποιων άλλων κόμβων της ιεραρχίας, θα αριθμηθούν με ids μικρότερα από τα ids των δεύτερων. Για παράδειγμα, αν θεωρήσουμε ότι το καθολικό σχήμα είναι αυτό του Σχ. 2.5, τότε οι κλάσεις θα αριθμούνταν ως εξής: C1→1, C4→2, C2→3, C5→4, C7→5, C3→6, C6→7 και C8→8. Έστω ότι το καθολικό σχήμα είναι ένα σχήμα S. Κάθε κλάση (τα αντίστοιχα θα ισχύουν για κάθε property του καθολικού σχήματος) ενός σχήματος θα παίρνει τον αριθμό-ταυτότητα της αντίστοιχης κλάσης του καθολικού σχήματος. Επίσης, σε κάθε κόμβο-κλάση του S αποθηκεύουμε ένα ζεύγος αριθμών: την ταυτότητα του κόμβου-κλάσης και την ταυτότητα του τελευταίου απογόνου του συγκεκριμένου κόμβου-κλάσης. Το ζεύγος που κρατάει η κάθε κλάση ενός σχήματος, θα το ονομάζουμε, *διάστημα* της κλάσης και θα το συμβολίζουμε ως [α, β].

Κάθε υποσχήμα του S περιέχει ένα αριθμό κλάσεων $C_0 \ldots C_k$ και τον αντίστοιχο αριθμό από properties $P_0 \ldots P_{k-1}$. Επομένως, το διάστημα ενός σχήματος αποτελείται από την ταυτότητα της κλάσης C_0 και από την ταυτότητα του τελευταίου απόγονου της κλάσης C_k. Επομένως, για να ελέγξουμε αν μια κλάση (ιδιότητα) C_1 είναι απόγονος μιας κλάσης (ιδιότητας) C_2, αρκεί να ελέγξουμε τα διαστήματα των δύο κλάσεων. Τη διαπίστωση αυτή, την αποτυπώνουμε με το Λήμμα 2.1.

*Λήμμα 2.*1. Μια κλάση (ιδιότητα) C_1 είναι απόγονος μιας κλάσης (ιδιότητας) C_2, αν και μόνο αν το διάστημα $[\alpha_1, \beta_1]$ της κλάσης (ιδιότητας) C_1 είναι υποσύνολο του διαστήματος $[\alpha_2, \beta_2]$ της κλάσης (ιδιότητας) C_2.

Απόδειξη: Έστω ότι η κλάση C_1 είναι απόγονος της κλάσης C_2. Σύμφωνα με τη pre-ordrer αρίθμηση που έχουμε επιβάλλει στο καθολικό σχήμα αν η κλάση C_1 είναι απόγονος της κλάσης C_2, τότε θα αριθμηθεί με id μεγαλύτερο από το id της κλάσης C_2, δηλαδή θα ισχύει $\alpha_1 > \alpha_2$ (1). Επίσης, ο τελευταίος απόγονος της κλάσης C_1 θα έχει id, ίδιο ή μικρότερο από τον τελευταίο απόγονο της κλάσης C_2. Αυτό ισχύει επειδή η pre-order αρίθμηση θα αριθμεί τα υποδένδρα της κλάσης C_2 από τα αριστερά προς τα δεξιά. Κατά συνέπεια, αν το υποδένδρο της κλάσης C_1 δεν είναι το πιο δεξιό, τότε ο τελευταίος απόγονος της C_1, θα έχει id μικρότερο από το id του τελευταίου απογόνου της C_2 (αφού η pre-order αρίθμηση θα αριθμήσει και άλλες κλάσεις, που είναι απόγονοι της C_2 και βρίσκονται σε υποδένδρα, πιο δεξιά από το υποδένδρο της C_1). Άρα θα ισχύει $\beta_1 \leq \beta_2$ (2). Από τις σχέσεις (1) και (2) έχουμε ότι $[\alpha_1, \beta_1]$ υποσύνολο του $[\alpha_2, \beta_2]$.

Αντιστρόφως: Έστω ότι $[\alpha_1, \beta_1]$ υποσύνολο του $[\alpha_2, \beta_2]$. Αυτό σημαίνει ότι $\alpha_1 \geq \alpha_2$ και $\beta_1 \leq \beta_2$. Αν $\alpha_1 = \alpha_2$, τότε σημαίνει πως μιλάμε για την ίδια κλάση, άρα $C_1 = C_2$. Αν $\alpha_1 > \alpha_2$, τότε σημαίνει πως η κλάση C_1 έχει αριθμηθεί μετά την κλάση C_2 (με βάση την pre-order αρίθμηση), άρα είτε είναι απόγονος της C_2 είτε βρίσκεται σε κάποιο δεξιότερο υποδένδρο (διαφορετικό) από την C_2. Αν βρίσκεται σε κάποιο δεξιότερο υποδένδρο, τότε ο τελευταίος απόγονος της C_1, θα είχε id μεγαλύτερο από τον τελευταίο απόγονο της κλάσης C_2. Άρα θα ίσχυε $\beta_1 > \beta_2$, πράγμα άτοπο, αφού από την υπόθεσή μας ισχύει $\beta_1 \leq \beta_2$. Άρα η κλάση C_1 είναι απόγονος της κλάσης C_2.

2.5.2. Υποθέσεις για τους Κόμβους του Συστήματος μας

Όπως είπαμε και πιο πάνω, κάθε κόμβος πρέπει να δημοσιοποιεί το RDF Σχήμα της βάσης μετά-δεδομένων του, το οποίο είναι υποσύνολο ενός καθολικού RDF Σχήματος. Ένας κόμβος μπορεί να εκδώσει περισσότερα από ένα σχήματα (για παράδειγμα, αν κρατάει δύο διαφορετικές βάσεις μετά-δεδομένων), αλλά, για να απλοποιήσουμε την περιγραφή του συστήματός μας, θα υποθέσουμε ότι κάθε κόμβος εκδίδει μόνο ένα σχήμα. Αυτό που, ουσιαστικά, αλλάζει στην περίπτωση που ένας κόμβος πρέπει να εκδώσει παραπάνω από ένα σχήμα, είναι οι χρόνοι εισαγωγής και

διαγραφής (αναλογικά με τον αριθμό των σχημάτων του κόμβου, πολλαπλασιάζεται και ο συγκεκριμένος χρόνος) του κόμβου. Πρέπει, όμως να τονίσουμε ότι, στην πραγματικότητα, μας ενδιαφέρουν τα σχήματα των κόμβων, αφού σκοπός μας είναι να δομήσουμε ένα σημασιολογικό δίκτυο κι αυτό μπορεί να συμβεί μόνο αν δομήσουμε, με κάποιο τρόπο, αυτά τα σχήματα. Επειδή, λοιπόν, στο σύστημά μας δεν μπορεί να γίνει εισαγωγή/διαγραφή ενός κόμβου χωρίς να εισαχθεί και το σχήμα της βάσης του κι επειδή υποθέτουμε ότι κάθε κόμβος εκδίδει μόνο ένα σχήμα, θα μιλάμε εναλλακτικά για εισαγωγή/διαγραφή σχήματος και όχι κόμβου. Έτσι, ο χρόνος εισαγωγής/διαγραφής του κόμβου, ταυτίζεται με τον χρόνο εισαγωγής/διαγραφής, του σχήματός του. Επίσης, η αναζήτηση ενός ερωτήματος, στην ουσία είναι αναζήτηση σχήματος (ενός ή περισσοτέρων κόμβων).

Όταν θα λέμε ότι ένα σχήμα συνδέεται με ένα άλλο σχήμα, θα εννοούμε ότι οι κόμβοι που εκδίδουν τα συγκεκριμένα σχήματα συνδέονται μεταξύ τους. Επίσης, όταν θα λέμε ότι τα σχήματα κρατάνε κάποιου είδους πληροφορία (για παράδειγμα, τα *fingertables*, όπως θα εξηγήσουμε και στη συνέχεια), θα εννοούμε ότι οι κόμβοι που εκδίδουν τα συγκεκριμένα σχήματα είναι αυτά που κρατάνε την πληροφορία. Ακόμη, όταν θα λέμε ότι τα σχήματα επιτελούν συγκεκριμένες λειτουργίες (όπως επικοινωνία με ένα άλλο σχήμα, προώθηση ενός ερωτήματος, ενημέρωση των *fingertables*, κλπ.), θα εννοούμε και πάλι, ότι οι κόμβοι είναι αυτοί που επιτελούν τις συγκεκριμένες λειτουργίες.

Ακόμη, έχουμε πει και πιο πάνω, ότι στο σύστημά μας δημιουργούνται ιεραρχίες σχημάτων (και πάλι να πούμε, ότι στην πραγματικότητα, δημιουργούνται ιεραρχίες κόμβων που εκδίδουν τα αντίστοιχα σχήματα). Όταν θα λέμε ότι ένα σχήμα είναι θυγατρικό ενός άλλου σχήματος, θα εννοούμε ότι τα δύο σχήματα έχουν συνδεθεί κατά τέτοιο τρόπο, στην ίδια ιεραρχία, ώστε το πρώτο σχήμα να είναι παιδί του δεύτερου. Αντίστοιχα, όταν θα λέμε ότι ένα σχήμα είναι πατρικό ενός δεύτερου, θα εννοούμε ότι τα δύο σχήματα έχουν συνδεθεί κατά τέτοιο τρόπο, στην ίδια ιεραρχία, ώστε το πρώτο να είναι πατέρας του δεύτερου. Επίσης, το σχήμα που βρίσκεται στην κορυφή κάθε ιεραρχίας, το ονομάζουμε υπερσχήμα, για να το ξεχωρίζουμε από τα άλλα σχήματα της ιεραρχίας (το σχήμα αυτό υπάγει όλα τα άλλα σχήματα της ίδιας ιεραρχίας). Είπαμε παραπάνω, ότι οι διαφορετικές ιεραρχίες των σχημάτων θα

δομούνται σε ένα δακτύλιο (παρόμοιο με αυτόν του Chord). Στην ουσία, πάνω στον δακτύλιο θα παίρνουν θέση μόνο τα υπερσχήματα των ιεραρχιών (και πάλι να πούμε, ότι στην πραγματικότητα, στον δακτύλιο δομούνται οι κόμβοι που εκδίδουν τα υπερσχήματα) και όχι κάποιος άλλος κόμβος της ιεραρχίας.

ΚΕΦΑΛΑΙΟ 3. ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ

3.1. Ορισμός Προβλήματος

Το πρόβλημα που εξετάζουμε με την παρούσα διατριβή είναι το εξής. Έστω ότι έχουμε ένα σύνολο από κόμβους οι οποίοι δημοσιοποιούν τις βάσεις μετά-δεδομένων τους με RDF Σχήματα. *Μπορούμε να δομήσουμε τα σχήματά τους κατά τέτοιο τρόπο, ώστε να φτιάξουμε ένα σημασιολογικό δίκτυο*; Πιο συγκεκριμένα, μας ενδιαφέρει να φτιάξουμε ιεραρχίες από σχήματα (ώστε περίπου ίδια σχήματα να βρίσκονται στην ίδια ομάδα σχημάτων), στις οποίες τα πατρικά σχήματα θα υπάγουν τα θυγατρικά σχήματα. *Μπορούμε, λοιπόν, να φτιάξουμε ένα σύστημα που να αποτελείται από τέτοιες ιεραρχίες*; Με άλλα λόγια, ενδιαφερόμαστε να τοποθετήσουμε κόμβους των οποίων τα σχήματα υπάγονται στο ίδιο σχήμα (δηλαδή, κόμβοι που είναι σημασιολογικά παρόμοιοι), στην ίδια ιεραρχία (δηλαδή, σε κοντινές θέσεις στο P2P δίκτυο). Ακόμη, *θα μπορούμε σε αυτό το σύστημα να μεταβούμε από μια ιεραρχία σε μια άλλη, σε μικρό σχετικά χρόνο, σε σχέση με το συνολικό αριθμό των ιεραρχιών*; Μας ενδιαφέρει τόσο η εισαγωγή/διαγραφή ενός σχήματος σε αυτό το σύστημα (άρα και η

εισαγωγή/διαγραφή του κόμβου που εκδίδει το συγκεκριμένο σχήμα), όσο και η αναζήτηση ενός ερωτήματος, να απαιτεί μικρό χρόνο (σε hops) και όσο το δυνατόν μικρό αριθμό μηνυμάτων, σε σχέση με τον συνολικό αριθμό των κόμβων του συστήματος. Επίσης, *θέλουμε ο χώρος που είναι αναγκασμένοι να κρατάνε οι κόμβοι (για να μπορεί να διατηρείται η δομή), να είναι λογαριθμικός, σε σχέση με το συνολικό αριθμό των κόμβων του δικτύου*. Στις επόμενες παραγράφους δίνουμε την απάντηση στα παραπάνω ερωτήματα.

3.2. Περιγραφή Τοπολογίας

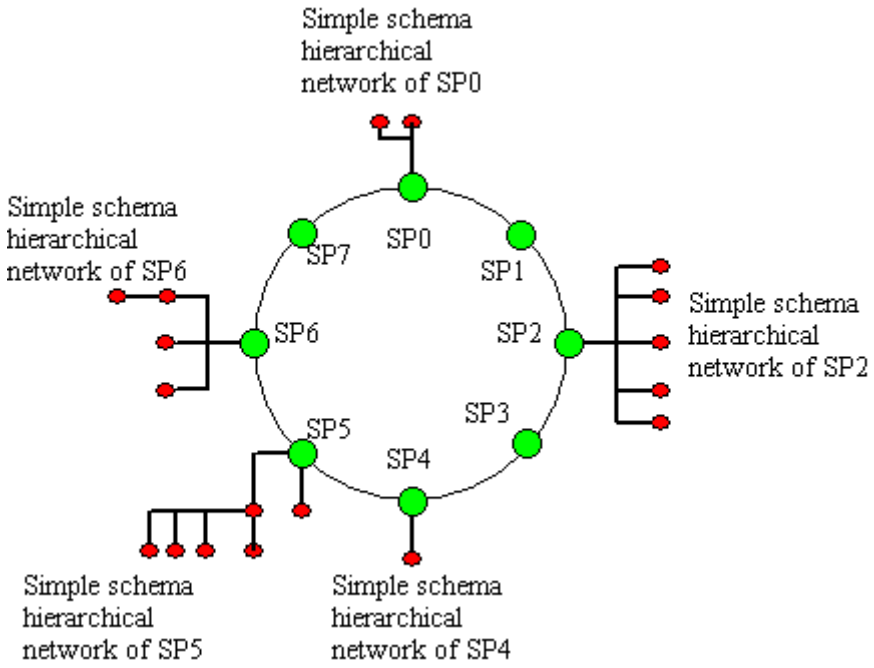
3.2.1. Περιγραφή Ιεραρχίας Σχημάτων

Όπως έχουμε πει, κάθε κόμβος εκδίδει ένα σχήμα. Τα σχήματα αυτά δομούνται σε ιεραρχίες. Κάθε ιεραρχία αποτελεί ένα ολοκληρωμένο σχήμα, το οποίο είναι διαφορετικό, από το ολοκληρωμένο σχήμα οποιασδήποτε άλλης ιεραρχίας. Το ριζικό σχήμα κάθε ιεραρχίας (το υπερσχήμα) περιέχει το πιο γενικό σχήμα, ενώ κάθε κόμβος της ιεραρχίας του, είναι, ουσιαστικά, ένα σχήμα που υπάγεται στο υπερσχήμα. Κάθε σχήμα S, σε μια ιεραρχία, μπορεί να έχει κάποια παιδιά (αν δεν είναι φύλλο στην ιεραρχία). Τα παιδιά του S, στην ουσία, είναι σχήματα που υπάγονται στο S. Επίσης, κάθε σχήμα S (εκτός από το υπερσχήμα), έχει και ένα πατρικό σχήμα στην ιεραρχία. Αυτό σημαίνει, ότι το S υπάγεται στο πατρικό του σχήμα. Κάθε σχήμα σε μια ιεραρχία, κρατάει και τα σχήματα των παιδιών του (αν έχει) καθώς και το σχήμα του πατέρα του (αν έχει). Επίσης, μερικά από τα υπερσχήματα μπορεί να έχουν κάποια κοινά στοιχεία (κλάσεις ή ιδιότητες κλάσεων). Αυτά τα υπερσχήματα δεν τα βάζουμε στην ίδια ιεραρχία, αλλά αποτελούν διαφορετικές ιεραρχίες (καταλαμβάνουν διαφορετικές θέσεις στον δακτύλιο, όπως θα πούμε παρακάτω). Πιο κάτω παραθέτουμε τους τυπικούς ορισμούς, που καθορίζουν πότε ένα σχήμα θα είναι παιδί ενός άλλου σχήματος και πότε θα είναι υπερσχήμα.

Ορισμός 3.1. Θα λέμε ότι ένα σχήμα S_1 είναι παιδί ενός σχήματος S_2 (S_1 είναι θυγατρικό σχήμα του S_2) ή το σχήμα S_2 είναι πατέρας του σχήματος S_1 (το S_2 είναι πατρικό σχήμα του S_1) αν και μόνο αν S_2 υπάγει το S_1.

Ορισμός 3.2. Θα λέμε ότι ένα σχήμα S είναι υπερσχήμα αν δεν υπάγεται από κανένα άλλο σχήμα.

Στο Σχ. 3.1, φαίνονται οι ιεραρχίες σχημάτων SP0, SP2, SP4, SP5 και SP6. Οι κύκλοι που βρίσκονται πάνω στο δακτύλιο είναι τα υπερσχήματα, ενώ οι μικρότεροι κύκλοι που βρίσκονται κάτω από τα υπερσχήματα αποτελούν τα σχήματα των ιεραρχιών.



Σχήμα 3.1 Τοπολογία Συστήματος

3.2.2. Περιγραφή Δακτυλίου Υπερσχημάτων

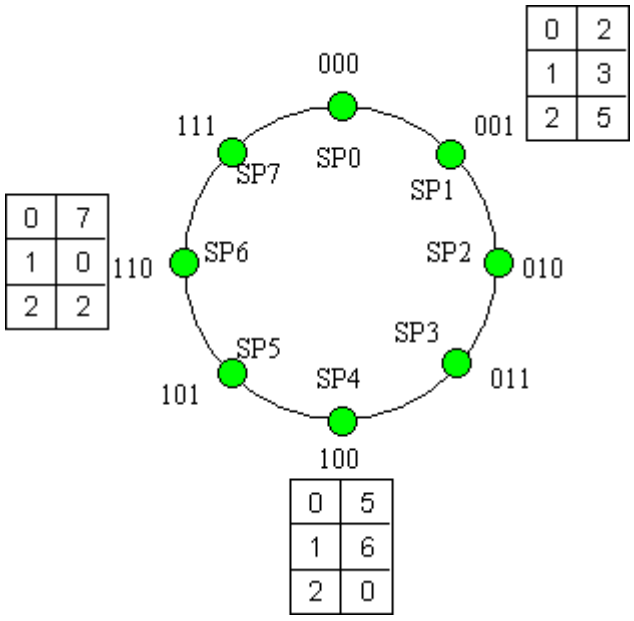
Κάθε υπερσχήμα δομείται σε ένα δακτύλιο, παρόμοιο με αυτόν του Chord [4]. Ο σκοπός αυτού του δακτυλίου είναι να χωρίσει σημασιολογικά το συνολικό δίκτυο και για αυτό το λόγο, κάθε ιεραρχία που σχηματίζεται κάτω από κάθε υπερσχήμα, πρέπει να αποτελεί ένα διαφορετικό σημασιολογικό δίκτυο.

Στο δακτύλιο αυτό, κάθε υπερσχήμα συνδέεται με O(⌈logn⌉) άλλα υπερσχήματα με ντετερμινιστικό τρόπο (όπως θα εξηγήσουμε και πιο κάτω), όπου n ο αριθμός των υπερσχημάτων (ουσιαστικά, ο αριθμός των ιεραρχιών) και ⌈logn⌉ το άνω ακέραιο μέρος του logn. Στη συνέχεια, για να απλοποιήσουμε την περιγραφή του συστήματός

μας, θα λέμε logn και θα αναφερόμαστε στο άνω ακέραιο μέρος του. Σε κάθε υπερσχήμα ανατίθεται ένα m-bit *αναγνωριστικό*, από μια συνάρτηση κατακερματισμού. Η συνάρτηση κατακερματισμού είναι τέτοια ώστε να διαμοιράζεται ο φόρτος σε όλα τα υπερσχήματα του συστήματος (ίδια με αυτή του Chord [4]). Για το αναγνωριστικό του υπερσχήματος, η συνάρτηση κατακερματισμού παίρνει ως είσοδο, την IP διεύθυνσή του κόμβου που εκδίδει το υπερσχήμα. Τα αναγνωριστικά των υπερσχημάτων οργανώνονται σε ένα κύκλο *modulo* 2^m. Τα υπερσχήματα τοποθετούνται σε δομή δακτυλίου (ο οποίος έχει αριθμημένες θέσεις από 0 - 2^m-1) σε αυτόν τον κύκλο, με βάση το αναγνωριστικό τους. Στο Σχ. 3.2, φαίνεται ένας δακτύλιος 8 θέσεων. Παρατηρούμε ότι κάθε θέση έχει αριθμηθεί με έναν m-bit αριθμό. Κάθε υπερσχήμα είναι συνδεδεμένο με το επόμενο (κατά τη φορά του ρολογιού) υπερσχήμα του δακτυλίου (*successor*) και με το προηγούμενο (*predecessor*). Επίσης, θα λέμε ότι το successor υπερσχήμα s, μιας θέσης K, του δακτυλίου, είναι το υπερσχήμα με αναγνωριστικό ίσο ή μεγαλύτερο του K και θα το συμβολίζουμε με s = successor(K). Αντίστοιχα, θα λέμε ότι το predecessor υπερσχήμα p, μιας θέσης K, του δακτυλίου, είναι το υπερσχήμα με αναγνωριστικό ίσο ή μικρότερο του K και θα το συμβολίζουμε με p = predecessor(K).

Για να διατηρηθεί αυτός ο δακτύλιος, θα πρέπει τα υπερσχήματα να διατηρούν τις δομές του δακτυλίου του Chord, που είναι γνωστές με το όνομα *fingertables*. Το fingertable είναι ένα πίνακας m θέσεων, αν έχουμε 2^m υπερσχήματα. Η i-οστή θέση του πίνακα ενός υπερσχήματος N, περιέχει το αναγνωριστικό ενός υπερσχήματος s, ο οποίος έπεται του N κατά 2^{i-1} τουλάχιστον, στο δακτύλιο των αναγνωριστικών, δηλαδή s = successor(N + 2^{i-1}) όπου $1 \leq i \leq m$ και όλη η αριθμητική είναι modulo 2^m. Με αυτόν τον τρόπο, κάθε υπερσχήμα χρειάζεται να γνωρίζει O(logn) άλλα υπερσχήματα, αν n είναι ο συνολικός αριθμός των υπερσχημάτων (για παράδειγμα, αν έχουμε 500 υπερσχήματα, τότε κάθε υπερσχήμα χρειάζεται να γνωρίζει O(log500) = O(9) άλλα υπερσχήματα). Στο Σχ. 3.2 φαίνονται, χαρακτηριστικά, τα fingertables των υπερσχημάτων 1, 4 και 6. Παρατηρούμε ότι κάθε υπερσχήμα έχει logn θέσεις στο fingertable του (για συντομία ονομάζονται *fingers*) και σε κάθε θέση, το υπερσχήμα που αποθηκεύεται, είναι το σχήμα που προκύπτει από τη σχέση s = successor(N + 2^{i-1}). Για παράδειγμα, η θέση 1 του υπερσχήματος 1 κρατάει το υπερσχήμα s = successor((1 + 2^{1-1}) mod 2^3) = 3, ενώ η θέση 3 του υπερσχήματος 6

κρατάει το υπερσχήμα s = successor((6 + 2^{3-1}) mod 2^3) = 2. Τα fingertables είναι απαραίτητα, γιατί θα μας βοηθήσουν, αργότερα, για να γίνει το broadcast ενός ερωτήματος ή ενός σχήματος στο δακτύλιο, σε O(logn) hops, αν n είναι το σύνολο των υπερσχημάτων.



Σχήμα 3.2 Δακτύλιος 8 θέσεων

Πρέπει να τονίσουμε εδώ, ότι εμείς, στο σύστημά μας δεν χρησιμοποιούμε κλειδιά, εν αντιθέσει με το Chord. Τα δεδομένα μας, που είναι τα RDF Σχήματα που εκδίδουν οι κόμβοι, δεν κατακερματίζονται από κάποια συνάρτηση κατακερματισμού, όπως αυτή του Chord που αναφέραμε παραπάνω. Κάθε σχήμα που εκδίδει ένας κόμβος βρίσκεται αποκλειστικά σε αυτόν τον κόμβο και δεν κατακερματίζεται σε κάποιον άλλο. Κατά συνέπεια, κανένας κόμβος δεν αποθηκεύει κάποιο σχήμα, πέρα από αυτό που εκδίδει (εν αντιθέσει με το chord, στο οποίο οι κόμβοι μπορεί να αποθήκευαν δεδομένα ή αναφορές σε δεδομένα, που δεν ανήκαν σε αυτούς). Αυτό το κάνουμε, γιατί έχουμε υποθέσει ότι κάθε κόμβος εκδίδει μόνο ένα σχήμα, οπότε δεν θα είχε νόημα να κατακερματίζουμε τα σχήματα. Αν τα κατακερματίζαμε, ίσως προέκυπταν κόμβοι που να ήταν υπεύθυνοι για αρκετά σχήματα, ενώ άλλοι, ενδεχομένως να μην ήταν υπεύθυνοι για κανένα. Ακόμη, όμως, και στην περίπτωση που οι κόμβοι εκδίδουν περισσότερα από ένα σχήματα, είναι λογικό να υποθέσουμε ότι δεν θα εκδίδουν πάρα πολλά (ίσως όχι περισσότερα από 3-4, αφού δεν περιμένουμε κάποιος

κόμβος να κρατάει περισσότερες από 3-4 διαφορετικές βάσεις μετά-δεδομένων), οπότε πάλι θα είναι φυσιολογικό κάποιοι κόμβοι να κρατάνε 3-4 σχήματα (οι κόμβοι που εκδίδουν αυτά τα σχήματα), ενώ κάποιοι άλλοι μόνο ένα (το σχήμα τους). Εκτός αυτού, θα ήταν δίκαιο οι κόμβοι που εκδίδουν πολλά σχήματα να είναι υπεύθυνοι και για αυτά και να δέχονται περισσότερο φόρτο από τους κόμβους που εκδίδουν μόνο ένα σχήμα.

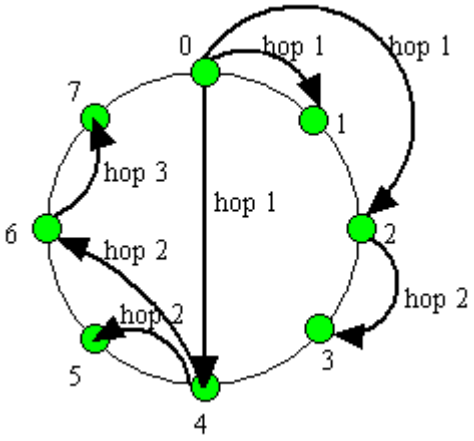
3.3. Εισαγωγή Σχήματος και Σύγκριση Σχημάτων

3.3.1. Εισαγωγή Σχήματος

Για την εισαγωγή ενός σχήματος στο σύστημα, δεν χρησιμοποιείται αποκλειστικά ο μηχανισμός του Chord. Ένα σχήμα που θέλει να εισαχθεί στο σύστημα, επικοινωνεί πρώτα με κάποιο υπερσχήμα. Το υπερσχήμα αυτό αναλαμβάνει να το κάνει broadcast στα υπόλοιπα υπερσχήματα στο δακτύλιο. Το broadcast στο δακτύλιο μπορεί να γίνει όπως το broadcast σε ένα δίκτυο Chord, σε O(logn) βήματα και n-1 μηνύματα, όπου n οι κόμβοι που ανήκουν στο δίκτυο Chord [18] (στην περίπτωσή μας n είναι ο αριθμός των υπερσχημάτων).

Πιο συγκεκριμένα, όταν ένα υπερσχήμα ξεκινήσει ένα broadcast, γίνονται οι εξής ενέργειες. Το μήνυμα broadcast περιέχει το σχήμα που γίνεται broadcast, το αναγνωριστικό του υπερσχήματος από όπου έχει σταλεί κι ένα *Όριο* που υπαγορεύει στο υπερσχήμα που λαμβάνει το broadcast, μέχρι ποιο (επόμενο) υπερσχήμα να το στείλει. Ένα συγκεκριμένο υπερσχήμα στέλνει το μήνυμα broadcast σε κάθε ένα από τα υπερσχήματα του fingertable του. Το όριο που θα στείλει μαζί με το μήνυμα είναι ίσο με το αναγνωριστικό της επόμενης θέσης του fingertable του. Ένα υπερσχήμα δεν στέλνει το μήνυμα broadcast σε όλα τα υπερσχήματα του fingertable του, αλλά μόνο σε αυτά, που το αναγνωριστικό τους δεν φτάνει το όριο που υποδυκνείεται στο broadcast μήνυμα (για την ακρίβεια, το αναγνωριστικό των υπερσχημάτων, στα οποία θα σταλεί το broadcast μήνυμα, θα πρέπει να βρίσκεται μεταξύ του αναγνωριστικού του υπερσχήματος στο οποίο βρισκόμαστε και στο όριο του μηνύματος). Με αυτόν τον τρόπο πετυχαίνουμε O(logn) hops και n-1 μηνύματα για το broadcast σε ένα δακτύλιο με n υπερσχήματα.

Στο Σχ. 3.3 φαίνεται ο τρόπος με τον οποίο κάνουμε broadcast στο σύστημά μας. Επίσης φαίνονται και τα συνολικά hops που απαιτούνται. Σε ένα δακτύλιο με 8 υπερσχήματα χρειαζόμαστε μόνο log8 hops, δηλαδή μόνο 3 hops. Έστω ότι το υπερσχήμα 0 ξεκινάει το broadcast. Το υπερσχήμα 0 θα στείλει το broadcast μήνυμα στα υπερσχήματα 1, 2 και 4 (όσα υπάρχουν στο fingertable του δηλαδή) με αντίστοιχα όρια 2, 4 και 0. Στη συνέχεια, το υπερσχήμα 1 δεν θα στείλει πουθενά το broadcast, γιατί κανένα από τα fingers του δεν ανήκει στο διάστημα (1, 2). Το υπερσχήμα 2 θα στείλει το broadcast μήνυμα μόνο στο υπερσχήμα 3, γιατί μόνο αυτό ανήκει στο διάστημα (2, 4). Αντίστοιχα, το υπερσχήμα 4 θα στείλει το μήνυμα στα υπερσχήματα 5 και 6 και το υπερσχήμα 6 θα στείλει το μήνυμα στο υπερσχήμα 7.



Σχήμα 3.3 Broadcast στο δακτύλιο του συστήματός μας

Κάθε υπερσχήμα που λαμβάνει το broadcast μήνυμα ελέγχει την σχέση υπαγωγής που έχει με το νεοεισερχόμενο σχήμα και απαντάει ανάλογα στο υπερσχήμα που εξέδωσε το broadcast. Δηλαδή, κάθε υπερσχήμα ελέγχει αν το νέο σχήμα υπάγει ή υπάγεται στο καινούριο σχήμα. Ο έλεγχος αυτός γίνεται με τον *Αλγόριθμο 3.2*, που περιγράφουμε στην υποπαράγραφο 3.3.2. Με βάση αυτό τον αλγόριθμο, εάν το νέο σχήμα δεν υπάγεται σε κάποιο υπερσχήμα και δεν υπάγει κάποιο υπερσχήμα, τότε το σχήμα παίρνει μια καινούρια θέση στο δακτύλιο. Σε αυτή την περίπτωση θα πρέπει ενημερωθούν τα fingertables των υπερσχημάτων του δακτυλίου. Η διαδικασία αυτή

είναι ίδια με αυτήν που περιγράφεται στο [4] για την ενημέρωση των fingertables των κόμβων του δακτυλίου του Chord.

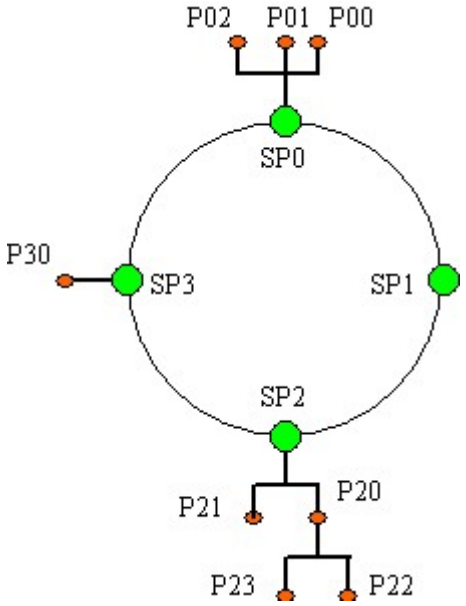
Εάν το νέο σχήμα S υπάγεται σε ένα ήδη υπάρχον υπερσχήμα S', τότε το S θα ενταχθεί στην ιεραρχία του S'. Εάν το νέο σχήμα S υπάγει ένα ή περισσότερα υπερσχήματα S_i, τότε το S θα πάρει μια θέση στο δακτύλιο, και τα υπερσχήματα S_i που υπάγει, θα διαγραφούν από τον δακτύλιο και θα ενωθούν κάτω από το υπερσχήμα που τα υπάγει, ως παιδιά του (οι ιεραρχίες που βρίσκονται κάτω από κάθε διαγραφέν υπερσχήμα, θα ακολουθήσουν τα αντίστοιχα υπερσχήματά τους). Αν ένα σχήμα S ανήκει σε περισσότερες από μια ιεραρχίες, τότε αυτό θα συνδεθεί με όλες τις ιεραρχίες (όταν θα θελήσει να διαγραφεί, θα πρέπει να διαγραφεί από όλες τις ιεραρχίες).

Όταν το νέο σχήμα φτάσει στο ριζικό σχήμα της ιεραρχίας (στο υπερσχήμα δηλαδή) με την οποία πρέπει να συνδεθεί, τότε γίνονται οι εξής ενέργειες. Το νέο σχήμα πρέπει να πάρει τη θέση του στην ιεραρχία, άρα ελέγχει το σχήμα του με το σχήμα των παιδιών του ριζικού σχήματος. Διακρίνουμε τρεις περιπτώσεις:

- Το νέο σχήμα υπάγεται σε ένα από τα παιδία του ριζικού σχήματος. Σε αυτή την περίπτωση, το νέο σχήμα κατεβαίνει ακόμη ένα επίπεδο στην ιεραρχία και η διαδικασία επαναλαμβάνεται επαναληπτικά.
- Το νέο σχήμα υπάγει ένα (ή περισσότερα) από τα παιδία του ριζικού σχήματος (φυσικά, τα ίδια θα ισχύουν αν το σχήμα δεν είναι το ριζικό, αλλά κάποιο σχήμα σε κατώτερο επίπεδο της ιεραρχίας). Σε αυτή την περίπτωση, το νέο σχήμα συνδέεται με το ριζικό σχήμα (ως παιδί τού ριζικού σχήματος) και με τα σχήματα που υπάγονται σε αυτό (τα σχήματα αυτά έχουν πλέον σαν πατρικό τους σχήμα, το νεοεισερχόμενο σχήμα).
- Το νέο σχήμα δεν υπάγεται ούτε υπάγει κάποιο από τα παιδιά του ριζικού σχήματος (τα ίδια θα ισχύουν αν το σχήμα δεν είναι ριζικό, αλλά κάποιο σχήμα σε κατώτερο επίπεδο της ιεραρχίας). Σε αυτή την περίπτωση, το νέο σχήμα ενώνεται με το ριζικό σχήμα (σαν παιδί του).

Παρακάτω, περιγράφουμε μερικά παραδείγματα που θα βοηθήσουν στην κατανόηση του μηχανισμού της εισαγωγής σχήματος.

Έστω, ότι αρχικά, ο δακτύλιος περιέχει τέσσερα υπερσχήματα, τα οποία είναι δομημένα όπως στο Σχ. 3.4. Τα τέσσερα αυτά υπερσχήματα περιγράφονται στον πίνακα 3.1.



Σχήμα 3.4 Σχήμα Δακτυλίου με Ιεραρχίες Σχημάτων

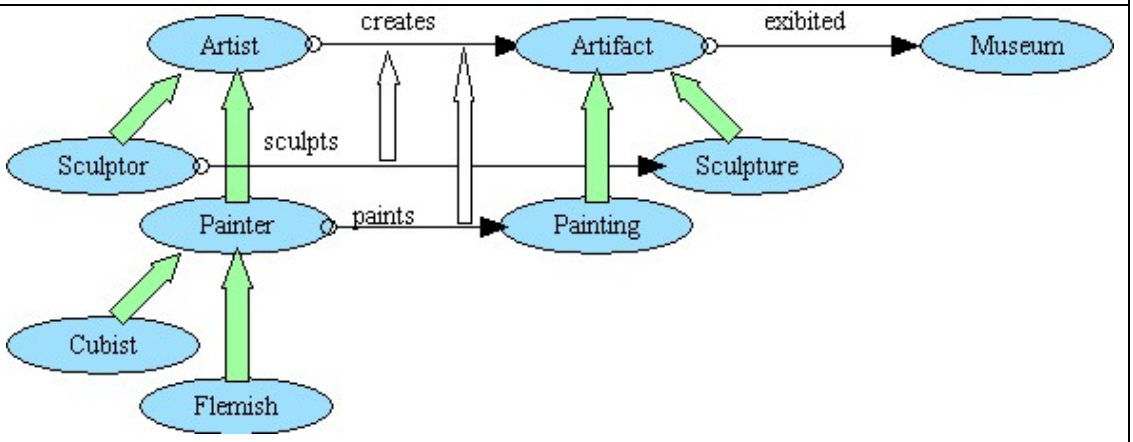
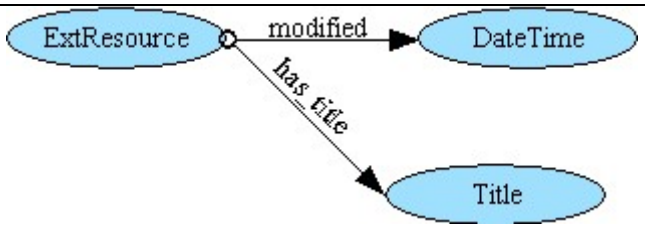
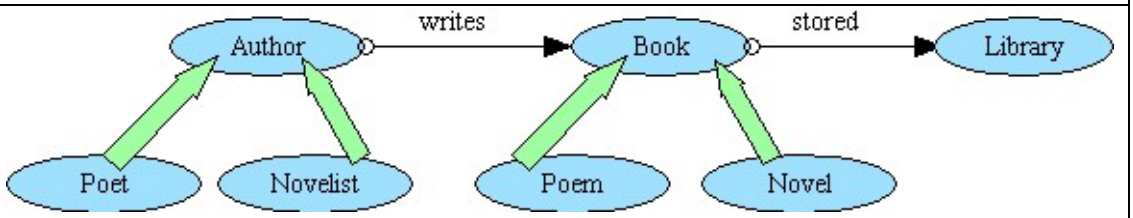
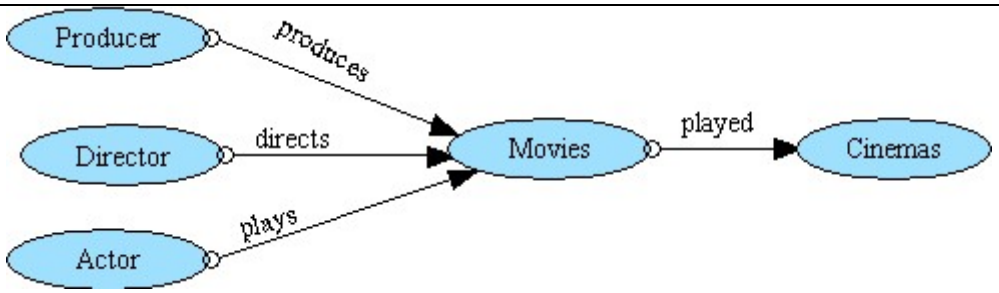
Έστω, ότι σε μερικές ιεραρχίες υπερσχημάτων υπάρχουν τα εξής σχήματα, τα οποία υπάγονται στα αντίστοιχα υπερσχήματα του πίνακα 3.1 (όπως φαίνεται και στο Σχ. 3.4):

Ιεραρχία του SP0: Το P00 περιέχει το σχήμα *Sculptor ----sculpts----> Sculpture*. Το P01 περιέχει το σχήμα *Painter ----paints----> Painting*. Το P02 περιέχει το σχήμα *Museum*.

Ιεραρχία του SP2: Το P20 περιέχει το σχήμα *Book ----stored----> Library*. Το P21 περιέχει το σχήμα *Poet ====> Author < ==== Novelist*. Το P22 περιέχει το σχήμα *Poem* και το P23 περιέχει το σχήμα *Novel*.

Ιεραρχία του SP3: Το P30 περιέχει το σχήμα *Actor ----plays----> Movies ---played--> Cinemas*

Πίνακας 3.1 Τα Σχήματα που Περιέχουν οι Υπερκόμβοι του Chord. Με Ελλείψεις Απεικονίζονται οι Κλάσεις, με Απλά Βέλη οι Ιδιότητες των Κλάσεων, με Χοντρά Σκούρα Βέλη οι Υποκλάσεις, ενώ με Χοντρά Λευκά Βέλη οι Υποϊδιότητες

Υπερσχήμα	*Σχήμα*
SP0: n0	
SP1: n1	
SP2: n2	
SP3: n3	

*Παράδειγμα 3.*1: Έστω, ότι θέλει να εισαχθεί στο σύστημα το σχήμα PX: *Director -directs----> Movies ---played----> Cinemas* και έστω, ότι αρχικά επικοινωνεί με το σχήμα P00. Η διαδικασία που θα ακολουθηθεί έχει ως εξής: Αρχικά, το σχήμα P00 θα παραπέμψει το νεοεισερχόμενο σχήμα, στο υπερσχήμα στου οποίου την ιεραρχία ανήκει, δηλαδή στο SP0. Το SP0, αφού ελέγξει αν το νέο σχήμα υπάγει ή υπάγεται

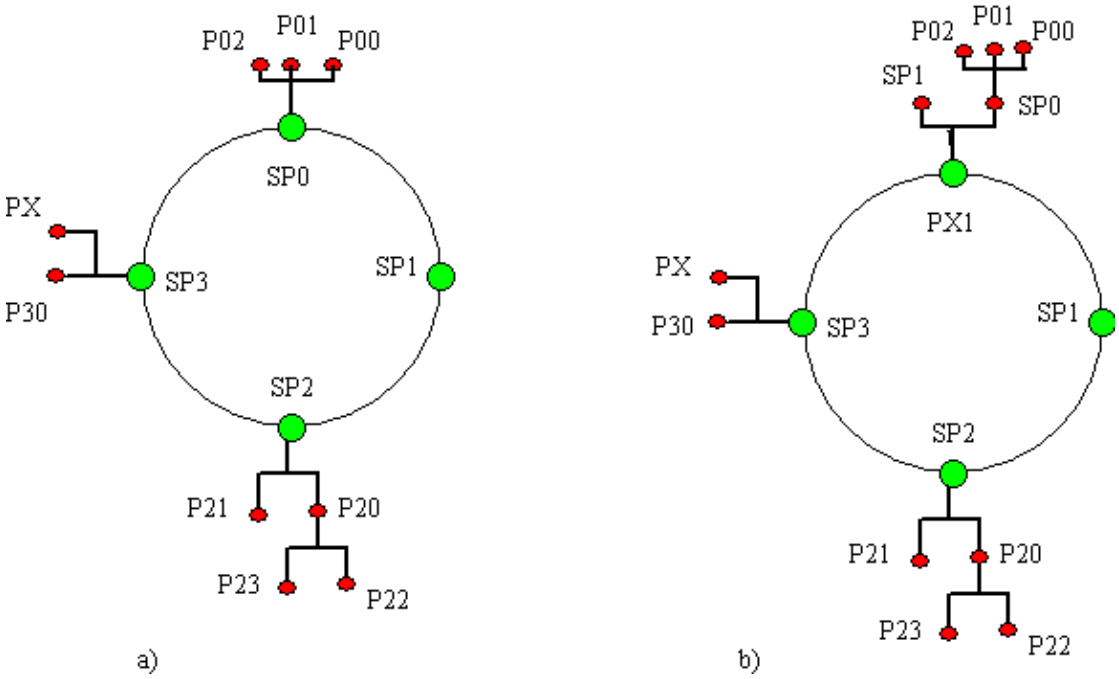
στο σχήμα του, το στέλνει με ένα broadcast μήνυμα, στα υπόλοιπα υπερσχήματα. Κάθε υπερσχήμα ελέγχει αν το σχήμα του υπάγει ή υπάγεται στο νέο σχήμα κι αν ναι, απαντάει αντίστοιχα στο SP0. Στην περίπτωσή μας, θα απαντήσει το υπερσχήμα SP3, το οποίο υπάγει το καινούριο σχήμα. Έτσι, το σχήμα PX θα επικοινωνήσει με το υπερσχήμα SP3. Το SP3 θα το προωθήσει στην ιεραρχία του, δηλαδή θα το προωθήσει στο σχήμα P30. Το P30 θα συγκρίνει το σχήμα του με το νέο σχήμα. Θα δει ότι το νέο σχήμα δεν υπάγεται ούτε υπάγει το σχήμα του P30, άρα θα συνδέσει το νέο σχήμα του PX στην ιεραρχία του, όπως φαίνεται στο Σχ. 3.5a.

Παράδειγμα 3.2: Ας θεωρήσουμε τώρα το εξής παράδειγμα. Έστω, ότι το σχήμα PX1 θέλει να εισαχθεί στο σύστημα. Το σχήμα PX1 φαίνεται στο Σχ. 3.6. Έστω, επίσης, ότι το νέο σχήμα επικοινωνεί με το σχήμα P22, άρα αυτό το παραπέμπει στο υπερσχήμα SP2. Το SP2 βλέπει ότι το σχήμα του δεν υπάγει, ούτε υπάγεται στο σχήμα PX1 και το στέλνει στα υπόλοιπα υπερσχήματα. Όταν πάρει τις απαντήσεις από τα υπόλοιπα υπερσχήματα, θα παρατηρήσει ότι το νέο σχήμα υπάγει τα υπερσχήματα SP0 και SP1. Άρα, αρχικά, θα διαγραφούν τα υπερσχήματα SP0 και SP1 από τον δακτύλιο (σε αυτό το σημείο θα πρέπει να ενημερωθούν τα fingertables των υπόλοιπων υπερσχημάτων για τη διαγραφή των SP0 και SP1), θα συνδεθούν με το νέο σχήμα PX1 σαν παιδιά του, ενώ το PX1 θα πάρει μια θέση στο δακτύλιο σαν υπερσχήμα (κι εδώ επίσης, θα ενημερωθούν τα fingertables των υπόλοιπων υπερσχημάτων, μετά την εισαγωγή του PX1 στο δακτύλιο), όπως φαίνεται στο Σχ. 3.5β.

Στο προηγούμενο παράδειγμα περιγράψαμε την εισαγωγή μιας κάθετης υπαγωγής (vertical subsumption). Παρόμοια, θα γινόταν η εισαγωγή μιας οριζόντιας υπαγωγής (horizontal subsumption).

Παράδειγμα 3.3: Για παράδειγμα, αν εισερχόταν στο σύστημα το σχήμα *Author—> writes--->Book----stored---->Library---based--->City*, το οποίο υπάγει οριζόντια το σχήμα SP2, τότε το υπερσχήμα SP2 θα διαγραφόταν από τον δακτύλιο, θα ενωνόταν με το καινούριο σχήμα, ως παιδί του και θα καταλάμβανε μια θέση στο δακτύλιο,.

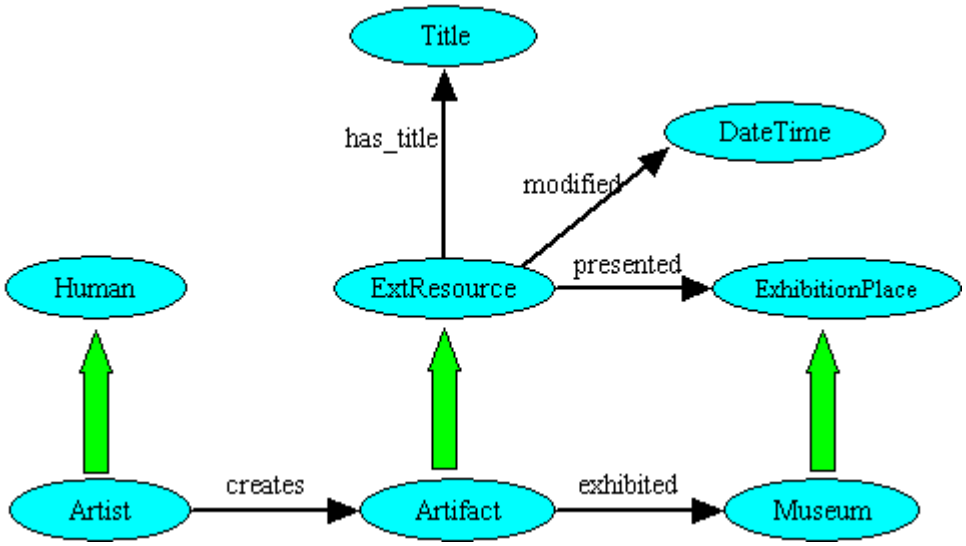
Παρατηρούμε, ότι το σύστημα που περιγράφουμε εξαρτάται από τη χρονική στιγμή που θα γίνουν οι εισαγωγές των σχημάτων. Για παράδειγμα, αν η εισαγωγή του σχήματος PX1 γινόταν πριν από την εισαγωγή των σχημάτων SP0 και SP1, τότε κατά την εισαγωγή των σχημάτων των SP0 και SP1, αυτά θα συνδεόταν στην ιεραρχία του PX1 (αφού τα σχήματά τους υπάγονται στο PX1), χωρίς να χρειαζόταν να γίνει κάποια διαγραφή υπερσχήματος (που κοστίζει σε hops και σε αριθμό μηνυμάτων).



Σχήμα 3.5 Το Σύστημά μας μετά την Εισαγωγή a) του PX, b) του PX1

Ο χρόνος που απαιτείται για να γίνει μια εισαγωγή ενός σχήματος στο σύστημα είναι p*O(logs) + O(h) hops (όπου s ο αριθμός των διαφορετικών υπερσχημάτων στον δακτύλιο, p ο αριθμός των υπερσχημάτων που πρέπει να διαγραφούν από τον δακτύλιο και h το ύψος της υψηλότερης ιεραρχίας του συστήματος), αφού χρειαζόμαστε O(logs) hops για να εντοπίσουμε τα υπερσχήματα (το οποία υπάγουν ή υπάγονται από το σχήμα που εισάγεται), p*O(logs) hops για τη διαγραφή των p υπερσχημάτων και O(h) για να εντοπίσουμε τη θέση του νέου σχήματος στην υψηλότερη ιεραρχία, στην οποία υπάγεται. Άρα, O(logs) + p*O(logs) + O(h) = p*O(logs) + O(h) hops. Επίσης, ο αριθμός των μηνυμάτων που απαιτούνται, είναι

O(s+k*h), αν k είναι ο αριθμός των ιεραρχιών στις οποίες υπάγεται το νεοεισερχόμενο σχήμα, αφού χρειαζόμαστε O(s) μηνύματα για το broadcast, p*O(logs) μηνύματα για τη διαγραφή των p υπερσχημάτων, O(h) μηνύματα για την εισαγωγή σε καθεμιά από τις k ιεραρχίες, και O(s) μηνύματα για τις απαντήσεις των υπερσχημάτων (στο broadcast). Άρα, χρειαζόμαστε O(s) + O(s) + p*O(logs) + O(k*h) = O(s+k*h) μηνύματα. Αυτός ο αριθμός των μηνυμάτων, είναι αυτός που απαιτείται στην χειρότερη περίπτωση, όπου το νεοεισερχόμενο σχήμα ανήκει σε παραπάνω από μία ιεραρχίες και υπάγεται σε πολλά από τα σχήματα κάθε ιεραρχίας. Ωστόσο, περιμένουμε ότι (στην συντριπτική πλειονότητα των περιπτώσεων) κάθε σχήμα θα ανήκει σε μια ιεραρχία και ότι θα υπάγεται σε ένα μόνο σχήμα αυτής της ιεραρχίας, οπότε το πλήθος των μηνυμάτων που απαιτείται σε αυτή την περίπτωση είναι O(s+h), αφού χρειαζόμαστε O(s) μηνύματα για το broadcast, p*O(logs) μηνύματα για τη διαγραφή των p υπερσχημάτων, O(h) μηνύματα για την εισαγωγή του νέου σχήματος στην ιεραρχία, και O(s) μηνύματα για τις απαντήσεις των υπερσχημάτων (στο broadcast). Άρα, χρειαζόμαστε O(s) + O(s) + p*O(logs) + O(h) = O(s+h) μηνύματα.



Σχήμα 3.6 Το Σχήμα του Κόμβου PX1

3.3.2. Αλγόριθμοι σύγκρισης Σχημάτων

Ας υποθέσουμε ότι έχουμε το εξής πρόβλημα: Έστω ένα σχήμα S και n υποσχήματα αυτού $S_0 \ldots S_{n-1}$. Ζητούμε να βρούμε i) αν S_i υπάγεται στο S_j, ii) αν S_j υπάγεται στο S_i, iii) αν $S_i = S_j$ όπου $S_i, S_j \varepsilon \{ S_0 \ldots S_{n-1} \}$.

Για να επιλύσουμε το παραπάνω πρόβλημα, εργαζόμαστε ως εξής. Με όσα είπαμε στο Κεφάλαιο 2, όλοι οι κόμβοι γνωρίζουν ένα καθολικό σχήμα και με βάση αυτό εκδίδουν τα σχήματά τους. Έστω S, αυτό το καθολικό σχήμα. Είπαμε επίσης, ότι κάθε κόμβος-κλάση του S κρατάει και το διάστημα [α, β] του κόμβου-κλάσης. Από τον ορισμό του RDF Σχήματος [1], κάθε σχήμα αποτελείται από ένα σύνολο από statements (ή τριάδες). Το σύνολο αυτό είναι υποσύνολο του συνόλου των statements του καθολικού σχήματος. Επίσης, κάθε statement ανήκει σε ένα επίπεδο, ανάλογα με το επίπεδο της ιδιότητας (κάθε ιδιότητα του καθολικού σχήματος ανήκει σε ένα επίπεδο, με βάση την καθολική ιεραρχία των ιδιοτήτων) την οποία περιέχει. Το επίπεδο των ιδιοτήτων μετράται από το ριζικό κόμβο-ιδιότητα της καθολικής ιεραρχίας κόμβων-ιδιοτήτων. Έτσι, για παράδειγμα, αν ένα statement ανήκει σε ένα επίπεδο μικρότερο από το επίπεδο ενός δεύτερου statement, τότε σίγουρα δεν υπάρχει περίπτωση το δεύτερο statement να περιέχει property, η οποία να είναι πρόγονος της property του πρώτου statement.

Με βάση τα όσα έχουμε περιγράψει μέχρι τώρα, μπορούμε να δώσουμε τον ορισμό υπαγωγής ενός statement st_1 σε ένα άλλο statement st_2:

Ορισμός 3.3: Έστω δύο statements st_1 και st_2. Θα λέμε ότι το st_1 υπάγει το st_2 (ή ότι το st_2 υπάγεται από το st_1), αν και μόνο αν (σύμφωνα με την ιεραρχία κλάσεων και ιδιοτήτων του καθολικού σχήματος):
i) η κλάση domain(st_2) είναι απόγονος της κλάσης domain(st_1) ή της κλάσης range(st_1) και
ii) η κλάση range(st_2) είναι απόγονος της κλάσης range(st_1) ή της κλάσης domain(st_1) και
iii) η ιδιότητα property(st_2) είναι απόγονος της ιδιότητας property(st_1).
Αν ισχύει ότι: domain(st_2) = domain(st_1) και range(st_2) = range(st_1) και property(st_2) = property(st_1), τότε θα λέμε ότι τα statements st_1 και st_2 είναι ίδια και θα συμβολίζουμε ως εξής: $st_1 = st_2$.

Με βάση τον Ορισμό 3.3 μπορούμε, τώρα, να ορίσουμε φορμαλιστικά την υπαγωγή ενός σχήματος S_1 σε ένα σχήμα S_2, καθώς και την οριζόντια και κάθετη υπαγωγή ενός σχήματος S_1 σε ένα σχήμα S_2.

Ορισμός 3.4: Έστω δύο σχήματα S_1 και S_2. Θα λέμε ότι το S_1 υπάγει το S_2 (ή ότι το S_2 υπάγεται από το S_1) αν και μόνο αν για κάθε statement st_j ε S_2 υπάρχει statement st_i ε S_1, ώστε το st_j υπάγεται στο st_i ή $st_j = st_i$. Ειδικά, αν για κάθε st_j ε S_2 υπάρχει st_i ε S_1 ώστε να ισχύει $st_j = st_i$ και $\Sigma st_j = \Sigma st_i$, τότε θα λέμε ότι τα S_1 και S_2 είναι ίδια και θα συμβολίζουμε ως εξής: $S_1 = S_2$.

Ορισμός 3.5: Έστω δύο σχήματα S_1 και S_2. Θα λέμε ότι το S_1 υπάγει οριζόντια το S_2 (ή ότι το S_2 υπάγεται οριζόντια από το S_1) αν και μόνο αν για κάθε statement st_j ε S_2 υπάρχει statement st_i ε S_1, ώστε $st_j = st_i$. Ειδικά, αν για κάθε st_j ε S_2 υπάρχει st_i ε S_1 ώστε να ισχύει $st_j = st_i$ και $\Sigma st_j = \Sigma st_i$, τότε θα λέμε ότι τα S_1 και S_2 είναι ίδια και θα συμβολίζουμε ως εξής: $S_1 = S_2$.

Ορισμός 3.6: Έστω δύο σχήματα S_1 και S_2. Θα λέμε ότι το S_1 υπάγει κάθετα το S_2 (ή ότι το S_2 υπάγεται κάθετα από το S_1) αν και μόνο αν για κάθε statement st_j ε S_2 υπάρχει statement st_i ε S_1, ώστε το st_j υπάγεται στο st_i.

Στη συνέχεια, παρουσιάζουμε 4 αλγορίθμους. Ο αλγόριθμος 3.4 χρησιμοποιείται από τον αλγόριθμο 3.2 (που είναι ο κύριος αλγόριθμος σύγκρισης σχημάτων), όταν χρειάζεται να ελέγξουμε αν ένα statement υπάγεται σε κάποιο άλλο statement ενός μόνο επιπέδου (όπως είπαμε και παραπάνω, κάθε statement ανήκει σε ένα επίπεδο, ίδιο με το επίπεδο που ανήκει η property του, με βάση την αρίθμηση της καθολικής ιεραρχίας των properties). Ο αλγόριθμος 3.1 χρησιμοποιείται από τους αλγορίθμους 3.3 και 3.4, όταν χρειαζόμαστε να συγκρίνουμε δύο statements. Επίσης, για να δουλέψει σωστά ο αλγόριθμος 2 θα πρέπει πρώτα τα statements των 2, προς σύγκριση, σχημάτων να είναι ταξινομημένα ως προς το επίπεδο που ανήκουν οι properties τους και ως προς την pre-order αρίθμηση, την οποία είχαμε επιβάλλει στο καθολικό σχήμα (δηλ. με βάση την ταυτότητα κάθε property). Για παράδειγμα, αν το διάστημα μιας property είναι [α, β], τότε η property θα ταξινομηθεί (στο επίπεδο το

Αλγόριθμος 3.1:

```
//Ο παρακάτω βοηθητικός αλγόριθμος συγκρίνει δύο statements st_i και st_j κι
//επιστρέφει 1 αν st_j υπάγεται στο st_i, 2 αν st_i υπάγεται στο st_j, 0 αν st_i =
//st_j και -1 σε κάθε άλλη περίπτωση.
//Έστω [domα_i, domβ_i]: διάστημα του domain του st_i, [ranα_i, ranβ_i]: διάστημα
//του range του st_i και [domα_j, domβ_j]: διάστημα του domain του st_j, [ranα_j,
//ranβ_j]: διάστημα του range του st_j.
//Έστω επίσης, [propα_i, propβ_i]: διάστημα της property του st_i και  [propα_j,
//propβ_j]: διάστημα της property του st_j.

1 int StatementSub(st_i, st_j)
2 begin
3 Αν: [domα_i, domβ_i] υποσύνολο του  [domα_j, domβ_j] και [ranα_i, ranβ_i]
4      υποσύνολο του  [ranα_j, ranβ_j] και [propα_i, propβ_i] υποσύνολο του
5      [propα_j, propβ_j]
6      ||
7      [domα_i, domβ_i] υποσύνολο του  [domα_j, domβ_j] και [ranα_i, ranβ_i]
8      υποσύνολο του [domα_j, domβ_j] και [propα_j, propβ_j] υποσύνολο του
9      [propα_i, propβ_i]
10     ||
11     [domα_i, domβ_i] υποσύνολο του  [ranα_j, ranβ_j] και [ranα_i, ranβ_i]
12     υποσύνολο του [ranα_j, ranβ_j] και [propα_i, propβ_i] υποσύνολο του
13     [propα_j, propβ_j]
14           Τότε: return 2;     //st_j υπάγει st_i
15 Αλλιώς αν: [domα_j, domβ_j] υποσύνολο του  [domα_i, domβ_i] και [ranα_j, ranβ_j]
16     υποσύνολο του  [ranα_i, ranβ_i] και [propα_j, propβ_j] υποσύνολο του
17     [propα_i, propβ_i]
14     ||
15     [domα_j, domβ_j] υποσύνολο του  [domα_i, domβ_i] και [ranα_j, ranβ_j]
16     υποσύνολο του [domα_i, domβ_i] και  [propα_j, propβ_j] υποσύνολο του
17     [propα_i, propβ_i]
17     ||
18     [domα_j, domβ_j] υποσύνολο του  [ranα_i, ranβ_i] και [ranα_j, ranβ_j]
19     υποσύνολο του [ranα_i, ranβ_i] και  [propα_j, propβ_j] υποσύνολο του
20     [propα_i, propβ_i]
21           Τότε: return 1;     //st_i υπάγει st_j
22 Αλλιώς αν: [domα_i, domβ_i] = =  [domα_j, domβ_j] και [ranα_i, ranβ_i] = =
23     [ranα_j, ranβ_j] και [propα_i, propβ_i] = = [propα_j, propβ_j]
24           Τότε: return 0;     //st_i = st_j
25 Αλλιώς
26     return -1;    //καμία σχέση υπαγωγής μεταξύ st_i, st_j
27 end
```

οποίο ανήκει), με βάση τον αριθμό-ταυτότητα α. Η δομή που χρησιμοποιούμε για να αποθηκεύσουμε τα statements των σχημάτων είναι η εξής. Αρχικά, αποθηκεύουμε τα statements ενός επιπέδου (αφού πρώτα, έχουμε ταξινομήσει τα statements με βάση το id της property κάθε statement) σε έναν πίνακα. Άρα, μέχρι στιγμής, έχουμε τόσους πίνακες, όσους και τα επίπεδα ενός σχήματος. Αυτούς τους πίνακες, τους ταξινομούμε ανά επίπεδο και τους αποθηκεύουμε και αυτούς σε ένα δυναμικό πίνακα. Έτσι λοιπόν, κάθε θέση αυτού του δυναμικού πίνακα θα έχει μια σειρά από ταξινομημένα statements που ανήκουν στο ίδιο επίπεδο.

Αλγόριθμος 3.2:

```
//Ο παρακάτω αλγόριθμος  συγκρίνει δύο σχήματα S1 και S2 επιστρέφει αν S2
//υπάγεται στο S1, 2 αν S1 υπάγεται στο S2, 0 αν S2 = S1 και -1 σε κάθε άλλη
//περίπτωση. Τα σχήματα δίνονται στον αλγόριθμο, σαν είσοδος, σε μορφή
//δυναμικού πίνακα, ο οποίος σε κάθε θέση του περιέχει μια σειρά από
//ταξινομημένα statements (τριάδες) του ίδιου επιπέδου. Τα statements
//είναι ταξινομημένα με βάση την property κάθε statement. Π.χ. αν [propa_i,
//propb_i] διάστημα μιας property, αυτή θα ταξινομηθεί με βάση το propb_i και
//όχι το propa_i.
//Με NumberOfLevels() υποδεικνύεται ο αριθμός των επιπέδων ιεραρχίας του
//αντίστοιχου σχήματος και με NumberOfStatements() υποδεικνύεται ο αριθμός
//των statements του αντίστοιχου επιπέδου. Με getStatement(i) παίρνουμε το
//statement υπ'  αριθμόν i του αντίστοιχου επιπέδου. Επίσης, με τη μέθοδο
//isSubsumedBy(level), ελέγχουμε αν ένα statement υπάγεται από το επίπεδο
//level.

1 int Subsumption(S1, S2):
2 begin
3 Αν: αρχικό επίπεδο του S1 μεγαλύτερο του αρχικού επιπέδου του S2
4      Τότε αν: Subsumption(S2, S1) = = 1
5             Τότε return 2;             //S1 υπάγεται στο S2
6      Αλλιώς αν: Subsumption(S2, S1) = = 2
7             Τότε return 1;             //S2 υπάγεται στο S1
8      Αλλιώς
9             return  Subsumption(S2, S1);
10 Αλλιώς:
11     level_i = 0;   // το τρέχον επίπεδο ιεραρχίας του S1
12     level_j = 0;   // το τρέχον επίπεδο ιεραρχίας του S2
13     st_i = 0;             // το τρέχον statement του level_i
14     st_j = 0;             // το τρέχον statement του level_j
15     while(level_j < S2.NumberOfLevels())
16            while(st_j < level_j.NumberOfStatements())
17                   while(level_i < S1.NumberOfLevels())
18                          Statement_j = level_j.getStatement(st_j);
19                          Sub = Statement_j.isSubsumedBy(level_i);
20                          Αν: Sub = = 0
21                                 Τότε αν: level_i = =  level_j
22                                        Τότε: return αδύνατο;
23                                 Αλλιώς: continue;
24                          Αλλιώς αν: Sub = = 1
25                                 Τότε: Προχώρα στο επόμενο statement st_j ;
26                                 level_i = 0 //συνέχισε το while του st_j
27                          Αλλιώς: return -1;  //καμία σχέση υπαγωγής
28                   end while
29                   Αν: είμαστε στο τελευταίο επίπεδο του S1
30                          Τότε: return -1;    //καμία σχέση υπαγωγής
31                   Αλλιώς: continue;
32            end while
33            Αν: είμαστε στο τελευταίο επίπεδο του S2
34                   Τότε: return 1             //S1 υπάγει S2;
35            Αλλιώς: continue;
36     end while
37 end
```

Ο αλγόριθμος 3.1 ελέγχει τη σχέση μεταξύ δύο statements (τριάδες). Για να ελέγξει αν μια κλάση C_1 (το ίδιο ισχύει για τις ιδιότητες των statements) είναι υποσύνολο μιας άλλης κλάσης C_2, ο αλγόριθμος ελέγχει αν το διάστημα της C_1 είναι υποσύνολο

του διαστήματος της C_2, σύμφωνα με το λήμμα 2.1 του Κεφαλαίου 2. Αν διαπιστώσει, ότι τα διαστήματα των κλάσεων ενός statement S_1 είναι υποσύνολα των κλάσεων ενός statement S_2 (ενδεχομένως οι κλάσεις του S_1 να είναι υποσύνολα μόνο μιας κλάσης του S_2) και ότι το διάστημα της ιδιότητας του S_1 είναι υποσύνολο του διαστήματος της ιδιότητας του S_2, τότε σύμφωνα με τον ορισμό 3.3, το statement S_1 υπάγεται στο statement S_2. Όπως παρατηρούμε, ο αλγόριθμος απαιτεί σταθερό χρόνο, αφού αρκούν 3 συγκρίσεις για να αποφανθούμε αν ένα statement υπάγεται σε ένα άλλο.

Ο αλγόριθμος 3.2 δουλεύει ως εξής. Ξεκινάει από τα επίπεδα των statements που βρίσκονται στο υψηλότερο επίπεδο. Ελέγχει αν κάθε statements του πρώτου επιπέδου του S_2 υπάγεται σε κάποιο από τα statements του επιπέδου $level_i$ του S_1 (αυτό γίνεται με τη μέθοδο isSubsumedBy(level), της οποίας τον αλγόριθμο θα περιγράψουμε στη συνέχεια). Αν βρει ότι ένα statement του S_2 υπάγεται σε κάποιο statement του συγκεκριμένου επιπέδου $level_i$ του S_1, τότε προχωράει στο επόμενο statement του επιπέδου $level_j$ του S_2. Αν τελειώσουν τα statements ενός συγκεκριμένου επιπέδου, τότε ο αλγόριθμος προχωράει στα statements του επόμενου επιπέδου. Αν βρει ότι κανένα statement του $level_i$ του S_1, δεν υπάγει το συγκεκριμένο statement, τότε ο αλγόριθμος προχωράει στο επόμενο επίπεδο $level_i$ του S_1, αρκεί το επόμενο επίπεδο του S_1 να μην είναι μικρότερο από το επίπεδο του S_2, γιατί σε αυτή την περίπτωση είμαστε σίγουροι ότι τα δύο σχήματα δεν έχουν σχέση υπαγωγής.

Η όλη διαδικασία τελειώνει, όταν βρούμε ότι όλα τα statements του S_2, υπάγονται σε κάποια, από τα statements του S_1 ή αν, ελέγχοντας κάποιο statement του S_2, φτάσουμε στο τέλος των statements του τελευταίου επιπέδου του S_1, χωρίς να βρούμε σχέση υπαγωγής. Έτσι, αν $number(st_i)$ και $number(st_j)$ το πλήθος των statements των σχημάτων S_1 και S_2 αντίστοιχα, τότε η πολυπλοκότητα του παραπάνω αλγορίθμου είναι $O(number(st_j)*\log^2(number(st_i)))$, αφού για κάθε statement st_j χρησιμοποιούμε τον αλγόριθμο isSubsumedBy(level), ο οποίος απαιτεί χρόνο O(logn). Στη χειρότερη περίπτωση, ο αλγόριθμος αυτός θα χρησιμοποιηθεί για κάθε επίπεδο του S_1 (συνολικά τα επίπεδα του S_1 είναι $\log(number(st_i))$). Ένας straightforward αλγόριθμος θα χρειαζόταν $O(number(st_i)* number(st_j))$ χρόνο. Στον Σχ. 3.3 παρουσιάζουμε έναν τέτοιο αλγόριθμο.

Αλγόριθμος 3.3:

```
//Ο αλγόριθμος αυτός είναι μια straightforward παραλλαγή του αλγορίθμου 3.2.
//Η είσοδος και έξοδος του παρακάτω αλγορίθμου είναι όμοια με αυτή του
//αλγορίθμου 2, όπως επίσης, και οι ονομασίες και οι συμβολισμοί.

1 int SFSubsumption(S1, S2):
2
3 begin
4 Αν: αρχικό επίπεδο του S1 μεγαλύτερο του αρχικού επιπέδου του Sj
5      Τότε αν: SFSubsumption (S2, S1) = = 1
6            Τότε return 2;            //S1 υπάγεται στο S2
7      Αλλιώς αν: SFSubsumption (S2, S1) = = 2
8            Τότε return 1;            //S2 υπάγεται στο S1
9      Αλλιώς
10           return  SFSubsumption (S2, S1);
11 Αλλιώς
12     Για κάθε Statementi του S1
13           Για κάθε Statementj του S2
14                 Αν: StatementSub(Statementi, Statementj) = = 1 ||
15                              StatementSub(Statementi, Statementj) = = 0
16                       Τότε continue;
17                 Αλλιώς
18                       break;
19           end loop
20
21           Αν: το Statementj είναι το τελευταίο statement του S1 και
22                       StatementSub(Statementi, Statementj) = = 1
23                 Τότε return 1;            //S2 υπάγεται στο S1
24           Αλλιώς αν: το Statementj είναι το τελευταίο statement του S1 και
25                       StatementSub(Statementi, Statementj) = = 0
26                 Τότε return 0;            //S2 = S1
27     end loop
28
29     return -1;    //καμία σχέση υπαγωγής
30 end
```

.

Αλγόριθμος 3.4:

```
//Η παρακάτω συνάρτηση χρησιμοποιείται σαν μέθοδος του Statement και ελέγχει
//αν το συγκεκριμένο statement υπάγεται στο επίπεδο (level) που δίνουμε σαν
//είσοδο. Επιστρέφει 1, αν το εκάστοτε statement υπάγεται από το level και 0
//αν το εκάστοτε statement δεν υπάγεται στο level.

1 int isSubsumedBy(level)
2 begin
3
4 Statementi =  ΔΔΑ(this, level); // το this είναι το συγκρινόμενο statement
5 Αν: StatementSub(Statementi, this) = = 1 || StatementSub(this, Statementi)
6           = = 0
7      Τότε: return 1;      //Το statement this υπάγεται στο επίπεδο level
8 Αλλιώς:
9      return 0;            //Το statement this δεν υπάγεται στο επίπεδο level
10
11 end
```

Ο αλγόριθμος 3.3 είναι μια ευθεία παραλλαγή του αλγορίθμου 3.2. Σε αυτήν την παραλλαγή, το κάθε statement του S_1 συγκρίνεται με όλα τα statements του S_2.

Επομένως, ο χρόνος που χρειάζεται ο αλγόριθμος αυτός για να αποφασίσει ποια είναι η σχέση υπαγωγής μεταξύ 2 σχημάτων είναι S_1 και S_2 είναι O(number(st_i)* number(st_j)).

Ο αλγόριθμος 3.4 ελέγχει αν ένα συγκεκριμένο statement (το this), υπάγεται σε κάποιο από τα statements του επιπέδου level. Ο αλγόριθμος ξεκινάει, κάνοντας μια δυαδική αναζήτηση (ΔΔΑ(this, level)), για να εξακριβώσει από ποιο statement πρέπει να ξεκινήσει η σύγκριση με το this. Είπαμε, ότι τα statements είναι ταξινομημένα σε κάθε επίπεδο, με βάση τα properties (πιο συγκεκριμένα, με βάση την ταυτότητα μιας συγκεκριμένης property). Επομένως, η δυαδική αναζήτηση ενός statement θα γίνει με τον ίδιο τρόπο. Δηλαδή, με βάση την ταυτότητα της property, του συγκεκριμένου statement (του this). Η δυαδική αναζήτηση επιστρέφει ένα statement. Το statement $Statement_i$ που επιστρέφει έχει property με ταυτότητα αμέσως μικρότερη (ή ίση) με την ταυτότητα της property του statement this. Με τον τρόπο αυτό δεν χρειάζεται να συγκρίνουμε το statement this με τα προηγούμενα statements του επιπέδου level (δηλ. με αυτά τα statements που είναι ταξινομημένα πριν το statement $Statement_i$). Στη συνέχεια, ο αλγόριθμος συγκρίνει το statement this με το $Statement_i$. Αν βρει ότι το $Statement_i$ υπάγει το this, επιστρέφει 1, αλλιώς επιστρέφει 0. Θεωρητικά, ο αλγόριθμος απαιτεί χρόνο O(logn), αν n είναι ο αριθμός των statements του επιπέδου level (όσο δηλαδή και ο χρόνος της δυαδικής αναζήτησης). Ωστόσο, αυτός ο χρόνος μπορεί να βελτιωθεί, αν κατά τη διάρκεια της δυαδικής αναζήτησης, παράλληλα, συγκρίνουμε και το statement this με τα statement που επισκεπτόμαστε κάθε φορά. Έτσι, ο χρόνος μειώνεται δραστικά, με αποτέλεσμα να μην χρειάζεται, σχεδόν ποτέ, χρόνος O(logn).

3.4. Διαγραφή Σχήματος

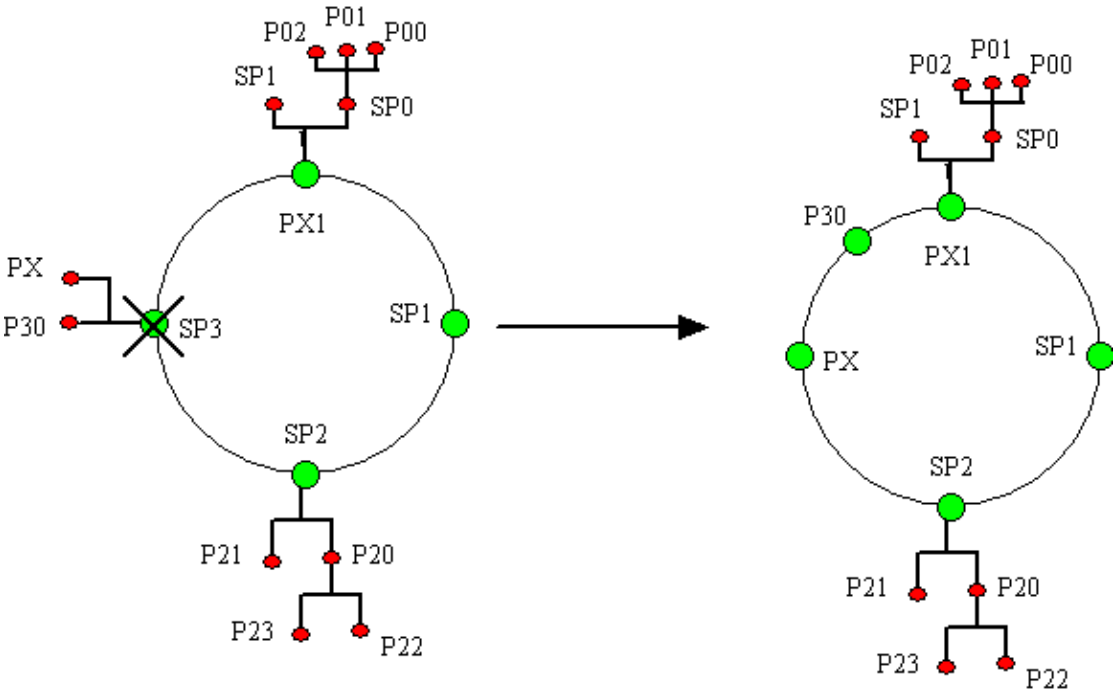
Για την περίπτωση της διαγραφής σχήματος, θεωρούμε δύο περιπτώσεις:
Διαγραφή υπερσχήματος από τον δακτύλιο. Στην περίπτωση αυτή, αρχικά θα πρέπει να ενημερωθούν τα fingertables των υπολοίπων υπερσχημάτων του δακτυλίου, για την αποχώρηση του συγκεκριμένου υπερσχήματος (οι αλγόριθμοι που ακολουθούνται για τη διαδικασία αυτή, είναι ίδιοι με την διαδικασία που απαιτείται, για την ενημέρωση

των fingertables του Chord [4], κατά την αποχώρηση ενός κόμβου). Στη συνέχεια, θα πρέπει όλα τα παιδιά του υπερσχήματος να ελέγξουν, αν ανήκουν σε κάποια άλλη ιεραρχία. Ο έλεγχος αυτός, μπορεί να γίνει εύκολα, γιατί κάθε σχήμα μιας ιεραρχίας κρατάει ένα δείκτη προς τον πατέρα του. Επομένως, αν ένα σχήμα έχει περισσότερους από ένα γονείς, που ανήκουν σε διαφορετικές ιεραρχίες, σημαίνει ότι και το ίδιο το σχήμα ανήκει σε δύο διαφορετικές ιεραρχίες. Άρα, αν ένα παιδί του υπερσχήματος που διαγράφηκε, διαπιστώσει ότι δεν ανήκει σε κάποια άλλη ιεραρχία, τότε θα πρέπει να πάρει μια θέση στο δακτύλιο, σαν υπερσχήμα και να ενημερώσει τα fingertables των άλλων υπερσχημάτων (η διαδικασία ενημέρωσης είναι ίδια με την περίπτωση της εισαγωγής ενός υπερσχήματος στο σύστημα). Αν ανήκει σε κάποια άλλη ιεραρχία, απλώς διαγράφει τον δείκτη προς το υπερσχήμα που διαγράφηκε (δηλαδή το σχήμα, τώρα, ανήκει μόνο σε μια ιεραρχία) και δεν απαιτείται καμία άλλη ενέργεια από το σύστημα. Εδώ, τονίζουμε ότι μόνο τα παιδιά του υπερσχήματος που αποχωρεί, εισάγονται στο δακτύλιο και όχι όλη η ιεραρχία του υπερσχήματος. Από την άλλη, οι ιεραρχίες που βρίσκονται κάτω από κάθε παιδί του υπερσχήματος, θα παραμείνουν και μετά την εισαγωγή των παιδιών στο δακτύλιο, κάτω από αυτά. Αν το υπερσχήμα που αποχωρεί δεν έχει παιδιά, τότε γίνονται μόνο οι διαδικασίες ενημέρωσης των fingertables.

Παράδειγμα 3.4: Για παράδειγμα, έστω ότι έχουμε το δίκτυο 3.5b, του παραδείγματος 3.2, και θέλουμε να διαγράψουμε το υπερσχήμα SP3, όπως φαίνεται στο Σχ. 3.11. Τα σχήματα PX, και P30, ελέγχουν αν ανήκουν σε κάποια άλλη ιεραρχία. Αυτό δεν ισχύει, άρα θα πρέπει να εισαχθούν στον δακτύλιο, κάτι το οποίο, θα τα οδηγήσει να γίνουν υπερσχήματα. Θα πρέπει να τονίσουμε, ότι αυτή η διαδικασία δεν θα γίνει για όλη την ιεραρχία του αποχωρήσαντος υπερσχήματος. Δηλαδή, δεν θα πρέπει όλοι οι κόμβοι της ιεραρχίας να εισαχθούν στο δακτύλιο, παρά μόνο, τα άμεσα παιδιά του αποχωρήσαντος υπερσχήματος. Τα σχήματα της υπόλοιπης ιεραρχίας (αν υπάρχει) δεν επηρεάζονται (αφού «κρέμονται» κάτω από την ιεραρχία των παιδιών του υπερσχήματος, θα ακολουθήσουν τα παιδιά του υπερσχήματος που διαγράφηκε).

Στην περίπτωση που μελετήσαμε μέχρι τώρα, ο χρόνος που απαιτείται είναι O(c*logs) hops (όπου c, ο αριθμός των παιδιών ενός υπερσχήματος που διαγράφεται, s το πλήθος των υπερσχημάτων και h το ύψος της βαθύτερης ιεραρχίας), αφού

χρειαζόμαστε O(logs) hops, για την διαγραφή ενός σχήματος και O(c*logs) hops για την εισαγωγή των c παιδιών (του υπερσχήματος που διαγράφεται) στον δακτύλιο (ουσιαστικά, για την ενημέρωση των fingertables των υπολοίπων υπερσχημάτων). Άρα, συνολικά, χρειαζόμαστε O(logs) + O(c*logs) = O(c*logs) hops. Επίσης, χρειαζόμαστε O(c*logs) αριθμό μηνυμάτων, αφού χρειαζόμαστε O(logs) μηνύματα για τη διαγραφή και O(c*logs) μηνύματα, για τις εισαγωγές των c παιδιών στον δακτύλιο (εδώ δεν χρειάζεται να κάνουμε broadcast τα σχήματα των παιδιών, παρά μόνο να ενημερώσουμε τα fingertables των υπόλοιπων υπερσχημάτων). Άρα, συνολικά χρειαζόμαστε O(logs) + O(c*logs) = O(c*logs) μηνύματα.

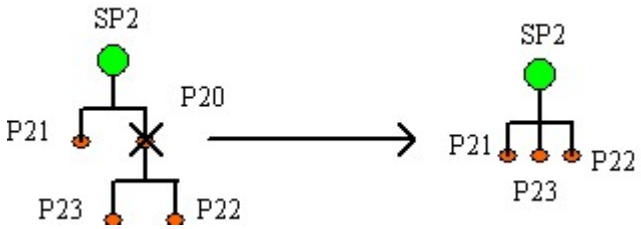


Σχήμα 3.7 Παράδειγμα Διαγραφής Υπερσχήματος

Διαγραφή σχήματος σε μια ιεραρχία. Στην περίπτωση που το σχήμα βρίσκεται εσωτερικά σε μια ιεραρχία, τότε η διαγραφή γίνεται εύκολα, σε ελάχιστο χρόνο. Εάν το σχήμα που φεύγει έχει παιδιά, τότε αυτά ενώνονται με τον πατρικό σχήμα τού αποχωρήσαντος σχήματος (για αυτό το λόγο, κάθε σχήμα πρέπει να κρατάει κι ένα δείκτη προς το πατρικό του σχήμα). Άρα, η διαδικασία της διαγραφής μπορεί να γίνει πολύ εύκολα, σε ένα βήμα.

Παράδειγμα 3.5. Για παράδειγμα, έστω ότι έχουμε το δίκτυο 3.5b, του παραδείγματος 3.2, και θέλουμε να διαγράψουμε το Σχήμα P20. Τότε, απλώς τα σχήματα P23 και P22, συνδέονται με το υπερσχήμα SP2, όπως φαίνεται στο Σχ. 3.12 (μέρος του σχήματος 3.5b).

Αν το σχήμα που αποχωρεί ανήκει σε περισσότερες από μια ιεραρχίες, τότε πρέπει τα παιδιά του να συνδεθούν με τους πατέρες του σχήματος από κάθε ιεραρχία.



Σχήμα 3.8 Διαγραφή Σχήματος σε Ιεραρχία

3.5. Αναζήτηση Ερωτήματος

Η αναζήτηση και απάντηση ενός ερωτήματος, μπορεί να γίνει παρόμοια με την εισαγωγή ενός σχήματος στο σύστημα. Έτσι, όταν ένα σχήμα δέχεται ένα ερώτημα, τότε το προωθεί προς το ριζικό του υπερσχήμα. Το ριζικό υπερσχήμα συγκρίνει το σχήμα του με το σχήμα του ερωτήματος (χρησιμοποιείται ο αλγόριθμος 3.2). Αν το υπερσχήμα του μπορεί να απαντήσει σε κάποιο μέρος του ερωτήματος, τότε το απαντά και προωθεί τα διαφορετικά μέρη του ερωτήματος προς τα παιδιά του και τα παιδιά του προς τα παιδιά τους κ.ο.κ. Οι απαντήσεις που παίρνει αυτό το ριζικό σχήμα από τα παιδιά του, συνενώνονται, έτσι ώστε να σχηματίσουν μια ολοκληρωμένη απάντηση. Παράλληλα, το ριζικό αυτό υπερσχήμα στέλνει το ερώτημα, προς όλα τα άλλα υπερσχήματα με ένα broadcast μήνυμα (παρόμοια με την εισαγωγή ενός σχήματος στο σύστημα). Τα υπερσχήματα του δακτυλίου, που λαμβάνουν αυτό το μήνυμα, ελέγχουν αν μπορούν να απαντήσουν σε κάποιο μέρος του ερωτήματος. Αν μπορούν να απαντήσουν, τότε απαντάνε και προωθούν το μέρος του ερωτήματος (στο οποίο μπορούν να απαντήσουν), στην ιεραρχία τους. Οι απαντήσεις, που θα πάρει το κάθε ριζικό σχήμα από τα παιδιά του, συνενώνονται και στέλνονται στο αρχικό ριζικό σχήμα, απ' όπου έχει προέλθει το μήνυμα broadcast.

Στη συνέχεια, συνενώνονται όλες οι επιμέρους απαντήσεις από τις υπόλοιπες ιεραρχίες, και στέλνονται (ως ολοκληρωμένη απάντηση) στον κόμβο, απ' όπου έχει προέλθει αρχικά το ερώτημα. Στη συνέχεια, περιγράφουμε μερικά παραδείγματα, για να κάνουμε κατανοητή την διαδικασία της αναζήτησης.

Παράδειγμα 3.5: Έστω, το δίκτυο του Σχ. 3.5b, στο οποίο οι κόμβοι περιέχουν ίδια σχήματα, με αυτά που περιγράψαμε στο παράδειγμα 3.1. Έστω ότι θέλουμε να απαντήσουμε στην εξής ερώτηση, την οποία κάνει ο κόμβος που εκδίδει το σχήμα P23: Βρείτε όλους τους καλλιτέχνες (Artist) που δημιουργούν καλλιτεχνήματα (Artifact) με ημερομηνία (Date) «2005». Αρχικά, η ερώτηση θα προωθηθεί προς το υπερσχήμα SP2. Στη συνέχεια, το σχήμα της ερώτησης θα συγκριθεί με το σχήμα SP2 και θα σταλεί στα υπόλοιπα υπερσχήματα (με broadcast). Ο SP2 δεν μπορεί να απαντήσει στην ερώτηση, ούτε ο SP3, άρα το μήνυμα δεν προωθείται στις ιεραρχίες τους. Στην ερώτηση μπορεί να απαντήσει το υπερσχήμα PX1. Το PX1 προωθεί το ερώτημα στην ιεραρχία του και συγκεκριμένα στα σχήματα SP0 και SP1. Και τα δύο αυτά σχήματα μπορούν να απαντήσουν σε μέρη του ερωτήματος (το SP1 μπορεί να απαντήσει με τα καλλιτεχνήματα που έχουν ημερομηνία «2005» και η ιεραρχία του SP0 στο υπόλοιπο ερώτημα). Το SP0 προωθεί το κομμάτι του ερωτήματος που μπορεί να απαντηθεί από την ιεραρχία του, στην ιεραρχία του. Οι απαντήσεις που παίρνει το SP0 από τα παιδιά του, συνενώνονται και στέλνονται στο SP2. Το SP1 δεν έχει ιεραρχία, άρα στέλνει τις δικές του απαντήσεις στο PX1. Το PX1 θα συνενώσει τις απαντήσεις που μπορεί να πάρει με τις δικές του και θα τις στείλει στο SP2. Το SP2 στη συνέχεια, θα συνενώσει όλα τα αποτελέσματα, που έχει λάβει από όλα τα υπερσχήματα και θα στείλει την τελική απάντηση στο P23.

*Παράδειγμα 3.*6: Έστω, τώρα, ότι ο κόμβος που εκδίδει το σχήμα PX, ρωτάει για τη δομή της ιεραρχίας του υπερσχήματος PX1. Το ερώτημα στέλνεται στο SP3 και αυτό το προωθεί στα υπόλοιπα υπερσχήματα του δακτυλίου. Το ερώτημα αναφέρεται μόνο στο υπερσχήμα PX1, άρα τα υπόλοιπα υπερσχήματα θα το αγνοήσουν και δεν θα το προωθήσουν στις ιεραρχίες τους. Το PX1 προωθεί το ερώτημα στα SP0 και SP1, μέσω της ιεραρχίας του (κι αυτά στις δικές τους ιεραρχίες) και το αποτέλεσμα που παίρνει, στέλνεται στο SP3. Αυτό, με τη σειρά του, στέλνει το αποτέλεσμα στον κόμβο απ' όπου προήλθε το ερώτημα (από τον κόμβο που εκδίδει το PX).

Παράδειγμα 3.7: Ας υποθέσουμε, τώρα, ότι ο κόμβος που εκδίδει το σχήμα P22 κάνει την ερώτηση: Βρείτε όλους τους συγγραφείς (Authors) και όλους τους παραγωγούς (Producers). Στην ερώτηση αυτή, μπορούν να απαντήσουν τα υπερσχήματα SP2 και SP3 και το σχήμα P21. Το ερώτημα, αρχικά στέλνεται στο SP3. το SP3 βλέπει ότι μπορεί να το απαντήσει, άρα το προωθεί στην ιεραρχία του, ενώ παράλληλα, το κάνει broadcast. Οι απαντήσεις που θα πάρει το SP2 από την ιεραρχία του (στη συγκεκριμένη περίπτωση, μόνο από το ίδιο το SP2 και από το P21) και από τα υπόλοιπα υπερσχήματα (από το υπερσχήμα SP3, συγκεκριμένα), θα ενωθούν και θα σταλούν στο P22.

Ο χρόνος που απαιτείται είναι O(logs) + O(h) hops (όπου s, ο αριθμός των διαφορετικών ιεραρχιών (υπερσχημάτων στο δακτύλιο) και h το ύψος της υψηλότερης ιεραρχίας), αφού χρειαζόμαστε O(logs) hops για το broadcast μήνυμα, O(h) hops για την αναζήτηση μέσα στην ιεραρχία και O(logs) hops για την απάντηση του κάθε ριζικού σχήματος (η απάντηση θα στέλνεται παράλληλα, από κάθε υπερσχήμα, άρα όσες ιεραρχίες και να απαντάνε στο ερώτημα, ο χρόνος θα παραμένει O(logs) hops). Άρα, έχουμε O(logs) + O(logs) + O(h) = O(logs) + O(h) hops. Ο αριθμός των μηνυμάτων που παράγονται είναι O(s) από το broadcast, O(s) από τις απαντήσεις των ριζικών σχημάτων και O(ν) για προώθηση του μηνύματος σε κάθε ιεραρχία (αν ν ο αριθμός κόμβων της μεγαλύτερης ιεραρχίας και με την υπόθεση ότι όλα τα υπερσχήματα του δακτυλίου και όλα τα σχήματα, όλων των ιεραρχιών απαντάνε σε κάποιο μέρος του ερωτήματος – χείριστη περίπτωση). Άρα, συνολικά έχουμε O(s) + O(s)*O(ν) = O(s*ν) μηνύματα. Άρα, ο συνολικός αριθμός μηνυμάτων θα είναι O(s*ν).

3.6. Εξισορρόπηση Φόρτου

Όσον αφορά την δίκαιη διασπορά των υπερσχημάτων στο δακτύλιο, αυτή επιτυγχάνεται, χρησιμοποιώντας την συνάρτηση κατακερματισμού SHA-1 (κατακερματίζουμε την IP του κόμβου, που εκδίδει το υπερσχήμα), για να τοποθετήσουμε ένα υπερσχήμα σε κάποια θέση στο δακτύλιο (στην ουσία, για να

δώσουμε στο υπερσχήμα το m-bit αναγνωριστικό, το οποίο αναφέραμε στην υποπαράγραφο 3.2.2).

Το σύστημα μας, όπως παρουσιάστηκε μέχρι τώρα, παρουσιάζει ένα αρκετά σημαντικό πρόβλημα, που είναι εγγενές στα σημασιολογικά δίκτυα κι αυτό είναι το πρόβλημα εξισορρόπησης φόρτου. Περιμένουμε, ότι κάποιες ιεραρχίες θα είναι πιο δημοφιλείς από κάποιες άλλες, δηλαδή σε λίγες ιεραρχίες θα υπάρχουν πολλά σχήματα, ενώ στις περισσότερες ιεραρχίες θα υπάρχουν λίγα σχήματα. Επίσης, περιμένουμε, ότι περισσότερα ερωτήματα θα απευθύνονται σε λίγες ιεραρχίες, ενώ ένας μικρός αριθμός ερωτημάτων, θα απευθύνονται στις υπόλοιπες ιεραρχίες. Έτσι, λίγες ιεραρχίες μπορούν να υπερφορτιστούν, προκαλώντας, κατά συνέπεια, υπερφόρτωση και σε όλο το σύστημα (bottleneck effect). Το πρόβλημα μπορεί να λυθεί, βάζοντας ένα όριο στον αριθμό των ερωτημάτων/μονάδα χρόνου, που μπορεί να δεχθεί (και να απαντήσει) μια ιεραρχία. Όταν εισάγουμε ένα νέο σχήμα σε μια ιεραρχία και αυτό το όριο ξεπεραστεί (στην επόμενη μονάδα χρόνου), τότε θα ήταν προτιμότερο να τοποθετήσουμε το σχήμα, που έχει εισαχθεί τελευταίο στην ιεραρχία, στον δακτύλιο (σαν υπερσχήμα) και να δημιουργήσουμε ένα υπερσύνδεσμο, από την ιεραρχία (στην οποία είχε εισαχθεί το σχήμα), προς το καινούριο αυτό σχήμα (αφού έχουμε διαπιστώσει ότι το πρώτο υπάγει το δεύτερο).

Το όριο του αριθμού των ερωτημάτων/μονάδα χρόνου, μπορεί να υπολογιστεί ως εξής. Έστω q_s, ο αριθμός των ερωτημάτων, που έχει απαντήσει η ιεραρχία ενός υπερσχήματος S, από τη στιγμή που εισήχθη το υπερσχήμα στο σύστημά μας και έστω, ότι από τη στιγμή της εισαγωγής έχει περάσει χρόνος T. Η μέση τιμή των ερωτημάτων/μονάδα χρόνου είναι $q_\mu = q_s/T$. Έστω επίσης, ότι ένα νέο σχήμα εισέρχεται στην ιεραρχία. Αν στην επόμενη μονάδα χρόνου ΔT, παρατηρηθεί (από το υπερσχήμα) ότι ο αριθμός των ερωτημάτων/μονάδα χρόνου, που απάντησε η ιεραρχία, είναι limit $\geq 2*q_\mu$, τότε θα ήταν καλύτερο να διαγράψουμε το νεοεισερχόμενο σχήμα από την ιεραρχία του S, να εισάγουμε το νέο σχήμα, σαν υπερσχήμα, στο δακτύλιο και να δημιουργήσουμε μια σύνδεση (έναν «υπερσύνδεσμο») μεταξύ των δύο υπερσχημάτων, για να επισημάνουμε τη σημασιολογική συσχέτισή τους. Με αυτή την ενέργεια, υποθέτουμε εμμέσως, ότι για τον επιπλέον φόρτο στην ιεραρχία του S, ευθύνεται το καινούριο σχήμα. Άρα, για να

αποφορτίσουμε την ιεραρχία, αποσυνδέουμε από αυτήν, το καινούριο σχήμα. Επιπλέον, έχουμε δημιουργήσει ένα υπερσχήμα με όριο limit = 2*q_{μ} (δηλαδή, όσο και το όριο της αρχικής ιεραρχίας, με την οποία είχε συνδεθεί το συγκεκριμένο σχήμα), δημιουργώντας έτσι δύο, περίπου εξισορροπημένες (όσον αφορά το φόρτο) ιεραρχίες σχημάτων.

Για να λειτουργήσει, όμως, αυτό το όριο που θέσαμε, θα πρέπει η χρονική μονάδα ΔΤ, με βάση την οποία μετράμε το q_{μ}, να είναι αρκετά μεγάλη, ώστε να προλάβει το σύστημα να σταθεροποιηθεί. Για παράδειγμα, όταν ο χρόνος Τ, για ένα συγκεκριμένο υπερσχήμα, είναι πολύ μικρός (δηλαδή, το υπερσχήμα έχει πολύ λίγο χρόνο που έχει εισαχθεί στο σύστημα), τότε ο αριθμός των ερωτημάτων που δέχεται, μπορεί να μεταβάλλεται κατά πολύ, ανάμεσα σε δύο διαδοχικά χρονικά διαστήματα, αν τα χρονικά διαστήματα είναι μικρά. Για το λόγο αυτό, χρειαζόμαστε αρκετά μεγάλα χρονικά διαστήματα ΔΤ, πάνω στα οποία θα μετράμε το q_{μ}. Από την άλλη, δεν μπορούμε να έχουμε υπερβολικά μεγάλα διαστήματα, γιατί τότε, μέχρι να μετρήσουμε το q_{μ} και μέχρι να ισορροπήσουμε το σύστημα, αυτό θα μένει για αρκετό χρόνο, μη ισορροπημένο. Ένας άλλος τρόπος για να λύσουμε το πρόβλημα των αρχικών στιγμών της εισαγωγής ενός υπερσχήματος, είναι να μην μετράμε το q_{μ} κάθε ΔΤ, αλλά κάθε μερικά ΔΤ (για παράδειγμα, κάθε 10 ΔΤ). Σε κάθε, όμως, διάστημα ΔΤ, θα βγάζουμε ένα limit με βάση το q_{μ}, καθενός ΔΤ, όπως περιγράψαμε παραπάνω. Στο τέλος, όλων των ΔΤ (π.χ. των 10), θα βγάζουμε ένα μέσο όρο, $limit_{\mu}$ όλων των limit που βρήκαμε σε καθένα ΔΤ. Σύμφωνα με αυτό το $limit_{\mu}$, θα αποφασίζουμε αν μια ιεραρχία είναι υπερφορτωμένη, οπότε και θα κάνουμε τις απαραίτητες ενέργειες για την εξισορρόπηση του φόρτου, στη συγκεκριμένη ιεραρχία.

ΚΕΦΑΛΑΙΟ 4. ΠΕΡΙΓΡΑΦΗ ΥΛΟΠΟΙΗΣΗΣ

4.1. Εισαγωγή

Στο κεφάλαιο αυτό, θα περιγράψουμε συνοπτικά, τον τρόπο με τον οποίο παράγαμε τα σχήματα των κόμβων και τον τρόπο, με τον οποίο υλοποιήσαμε τα τρία συστήματα, τα οποία συγκρίνουμε στο Κεφάλαιο 5. Πιο συγκεκριμένα, στην παράγραφο 4.2, περιγράφουμε τον τρόπο, με τον οποίο παράγουμε τα σχήματα, τα οποία εκδίδουν οι κόμβοι στα τρία συστήματα. Στις παραγράφους 4.3 και 4.4, περιγράφουμε τον τρόπο με τον οποίο έχουμε υλοποιήσει το σύστημά μας και το αδόμητο σύστημα, αντιστοίχως. Στην παράγραφο 4.5, περιγράφουμε το σύστημα [12], αλλά και τον τρόπο που το έχουμε υλοποιήσει, προκειμένου να το συγκρίνουμε, πειραματικά, με το δικό μας σύστημα.

4.2. Παραγωγή Σχημάτων

Για την υλοποίηση του συστήματός μας χρησιμοποιήσαμε τη γλώσσα Java. Για την παραγωγή του καθολικού σχήματος χρησιμοποιήσαμε την γεννήτρια RDF Σχημάτων RBench [19] του F.O.R.T.H.

Ο RBench είναι μία γεννήτρια RDF Σχημάτων. Για την ακρίβεια, αυτό που κάνει ο RBench είναι να δημιουργεί μια ιεραρχία από κλάσεις (ή αντίστοιχα από properties), με συγκεκριμένο βάθος και fan-out 2. Σε αυτήν την ιεραρχία κλάσεων (αντιστοίχως properties), όταν μια κλάση Α είναι παιδί μιας άλλης κλάσης Β, υπονοείται ότι η πρώτη υπάγεται από τη δεύτερη (δηλαδή συνδέονται μεταξύ τους με μια σχέση isA που ξεκινάει από την Α και καταλήγει στην Β). Στη συνέχεια, ο RBench αντιστοιχίζει σε όλες οι κλάσεις ένα σύνολο από στιγμιότυπα. Η αντιστοίχηση αυτή, δεν γίνεται απαραίτητα ομοιόμορφα, αλλά με κάποια κατανομή, που μπορεί να δοθεί στον RBench, ως παράμετρος.

Από το RBench, χρησιμοποιήσαμε το κομμάτι του κώδικα, με τον οποίο παράγεται η ιεραρχία των κλάσεων. Το συγκεκριμένο κομμάτι κώδικα, το χρησιμοποιούμε για την παραγωγή μιας καθολικής ιεραρχίας κλάσεων. Επίσης, αυτό το κομμάτι κώδικα χρησιμοποιείται και για την παραγωγή μιας αντίστοιχης, καθολικής ιεραρχίας properties. Στη συνέχεια, οι properties αντιστοιχίζονται στην καθολική ιεραρχία κλάσεων, που έχουμε ήδη παράξει, δημιουργώντας έτσι, ένα καθολικό σχήμα. Με βάση αυτό το καθολικό σχήμα, παράγουμε διάφορα υποσχήματα τα οποία εκδίδονται, εικονικά, από τους κόμβους που θέλουν να εισαχθούν στο σύστημά μας.

Ο κώδικας παραγωγής ιεραρχίας κλάσεων του RBench, παράγει ιεραρχίες, όπου κάθε κόμβος περιέχει δύο παιδιά (μέχρι ένα συγκεκριμένο βάθος). Ωστόσο, τροποποιώντας κατάλληλα τον κώδικα, μπορούμε να παράξουμε ιεραρχίες (από κλάσεις ή properties), ώστε κάθε κόμβος να έχει διαφορετικό αριθμό από παιδιά.

4.3. Υλοποίηση του Συστήματός μας

Για να παραστήσουμε ένα σχήμα, δημιουργήσαμε την κλάση Node που, εκτός των άλλων, περιέχει το καθολικό Σχήμα, το σχήμα του συγκεκριμένου κόμβου, αλλά και τον αλγόριθμο σύγκρισης δύο σχημάτων. Κάθε Σχήμα στο σύστημα μας, παριστάνεται με ένα στιγμιότυπο αυτής της κλάσης. Το Σχήμα που αντιστοιχίζουμε σε κάθε στιγμιότυπο της κλάσης Node, το αποθηκεύουμε στον Node ανά επίπεδο,

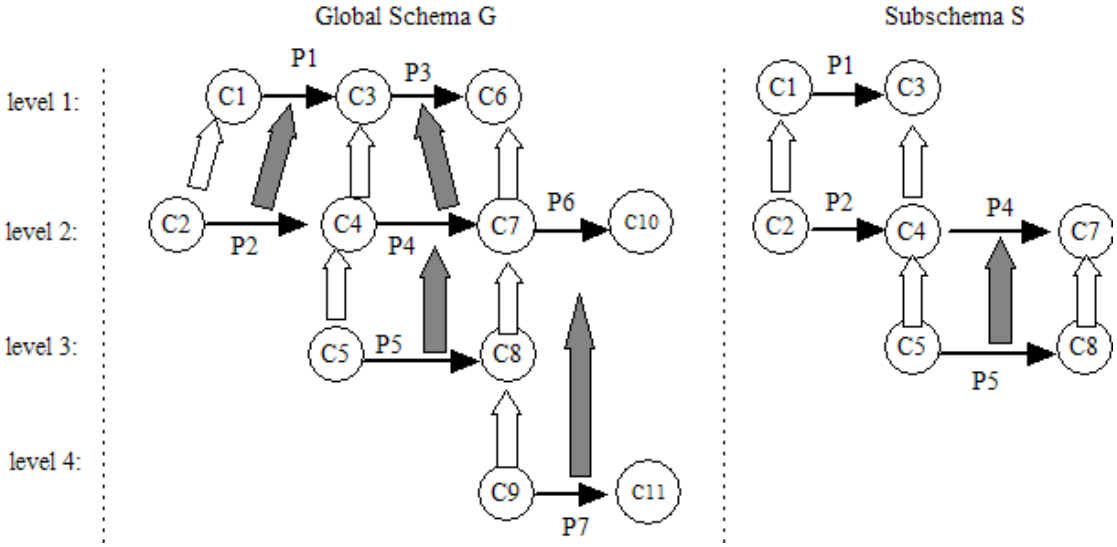
δηλαδή αποθηκεύουμε τις τριάδες του επιπέδου i όλες μαζί, τις τριάδες του επιπέδου i+1, επίσης όλες μαζί κ.ο.κ. Αυτό γίνεται, έτσι ώστε να μπορούμε να συγκρίνουμε δύο σχήματα ανά επίπεδο (κάτι που χρειάζεται στον αλγόριθμο 3.2).

Παράδειγμα 4.1

Για παράδειγμα, στο Σχ. 4.1, φαίνεται ένα καθολικό RDF Σχήμα G και ένα υποσχήμα του καθολικού σχήματος, S. Με λευκά βέλη παριστάνονται οι σχέσεις isA μεταξύ των κλάσεων, ενώ με σκούρα βέλη παριστάνονται οι σχέσεις isA μεταξύ των ιδιοτήτων. Παρατηρούμε, ότι τόσο οι κλάσεις του καθολικού σχήματος, όσο και οι ιδιότητες έχουν αριθμηθεί με pre-order αρίθμηση. Επίσης, παρατηρούμε ότι έχει δημιουργηθεί μια ιεραρχία από κλάσεις και από ιδιότητες, σε κάθε επίπεδο της οποίας, αντιστοιχούν κάποιες κλάσεις και κάποιες ιδιότητες. Έτσι, στο επίπεδο 1 αντιστοιχούν οι κλάσεις C1, C3, C6 και οι ιδιότητες P1, P3. Στο επίπεδο 2 αντιστοιχούν οι κλάσεις C2, C4, C7, C10 και οι ιδιότητες P2, P4, P6. Στο επίπεδο 3 αντιστοιχούν οι κλάσεις C5, C8 και η ιδιότητα P5. Στο επίπεδο 4, ανήκουν οι κλάσεις C9, C11 και η ιδιότητα P7. Ένας κόμβος που θα εκδώσει ένα σχήμα, το οποίο είναι υποσχήμα του καθολικού σχήματος, θα το αποθηκεύσει ως εξής. Θα αποθηκεύσει τις τριάδες ανά επίπεδο, με βάση το επίπεδο, στο οποίο ανήκει η ιδιότητα της τριάδας. Επίσης, σε κάθε επίπεδο θα έχει τις τριάδες διαταγμένες, με βάση την ιδιότητα της τριάδας. Έτσι, ο κόμβος που θέλει να εκδώσει το υποσχήμα S, θα αποθηκεύσει στο επίπεδο 1, την τριάδα C1→C3, στο επίπεδο 2, κατά σειρά τις τριάδες C2→C4 και C4→C7 (αφού τις δύο τριάδες, τις συνδέουν οι ιδιότητες P2 και P4, αντίστοιχα) και στο επίπεδο 3, την τριάδα C5→C8.

Η σύγκριση του σχήματος ενός στιγμιοτύπου της κλάσης Node που ήδη υπάρχει στο σύστημα, με το σχήμα ενός νέου στιγμιοτύπου της ίδιας κλάσης, που θέλει να εισαχθεί στο σύστημα, έχει τις εξής συνέπειες. Είτε το νέο στιγμιότυπο θα προσαρτηθεί κάτω από το «παλιό» σαν παιδί του, είτε θα προωθηθεί σε ένα από τα παιδιά του (για να γίνει καινούρια σύγκριση σχημάτων), είτε θα γίνει πατρικό στιγμιότυπο του «παλιού», είτε θα φύγει εντελώς από την ιεραρχία του «παλιού» στιγμιοτύπου (στην περίπτωση, που το «παλιό» στιγμιότυπο είναι ριζικός κόμβος μιας ιεραρχίας και τα σχήματα των δύο κόμβων είναι διαφορετικά). Στην περίπτωση που το σχήμα ενός «νέου» στιγμιοτύπου υπάγει τα σχήματα ενός η περισσοτέρων

στιγμιοτύπων, που παριστάνουν υπερσχήματα, τότε το «νέο» στιγμιότυπο παίρνει τη θέση του στον δακτύλιο και τα εν λόγω στιγμιότυπα αποχωρίζονται από τις ιεραρχίες τους και ενώνονται κάτω από το καινούριο στιγμιότυπο, σαν παιδιά του. Σε κάθε περίπτωση, το γενικότερο αποτέλεσμα είναι να δημιουργηθούν ιεραρχίες από στιγμιότυπα, τοποθετημένα με βάση τα σχήματά τους, όπως περιγράφεται στο μοντέλο μας (Κεφάλαιο 3). Τα στιγμιότυπα που παριστάνουν υπερσχήματα τοποθετούνται στο δακτύλιο, με βάση τους αλγορίθμους που περιγράφονται στο [4] (όσον αφορά την αρχικοποίηση και την ενημέρωση των fingertables των υπερσχημάτων).



Σχήμα 4.1 Επίπεδα RDF Σχημάτων

Πρέπει να τονίσουμε, ότι για τη δημιουργία του δακτυλίου στο σύστημά μας, χρησιμοποιήσαμε τους αλγορίθμους που αναφέρονται στο [4], οι οποίοι δίνουν $O(\log^2 n)$ πολυπλοκότητα, σε hops και πλήθος μηνυμάτων (όπου n ο αριθμός των κόμβων του δακτυλίου του chord), τόσο για την εισαγωγή κόμβου, όσο και για τη διαγραφή.

4.4. Υλοποίηση του Αδόμητου Συστήματος

Για τα πειράματά μας δημιουργήσαμε κι ένα δίκτυο, στο οποίο οι κόμβοι συνδέονται τυχαία μεταξύ τους, χωρίς να συγκροτούν κάποια συγκεκριμένη δομή, ούτε κάποια συγκεκριμένη ιεραρχία. Για την υλοποίηση αυτού του συστήματος, δανειστήκαμε την κλάση Node του προηγούμενου συστήματος, αλλάξαμε, όμως, τις διαδικασίες εισαγωγής/ διαγραφής σχήματος και αναζήτησης ερωτήματος. Εδώ, ένας κόμβος που θέλει να εισαχθεί στο σύστημα, συνδέεται τυχαία με κάποιον άλλο κόμβο του συστήματος, κάτι που μπορεί να γίνει σε ένα hop, με την ανταλλαγή ενός μηνύματος. Το ίδιο ισχύει για την περίπτωση της διαγραφής. Όταν ένας κόμβος φεύγει από το σύστημα δεν χρειάζεται να ενημερώσει κάποιον άλλο, άρα και η διαδικασία της διαγραφής μπορεί να γίνει τετριμμένα, σε σταθερό χρόνο. Στην περίπτωση, όμως, της αναζήτησης ερωτήματος, το ερώτημα θα πρέπει να περάσει από όλους τους κόμβους, για να διαπιστώσει, αν μπορεί να απαντηθεί από αυτούς, κάτι που μπορεί να γίνει με τη μέθοδο της πλημμύρας, σε χρόνο $\Theta(n)$ hops και $\Omega(n)$ πλήθος μηνυμάτων, αν n είναι το πλήθος των κόμβων του συστήματος.

4.5. Περιγραφή και Υλοποίηση του [12]

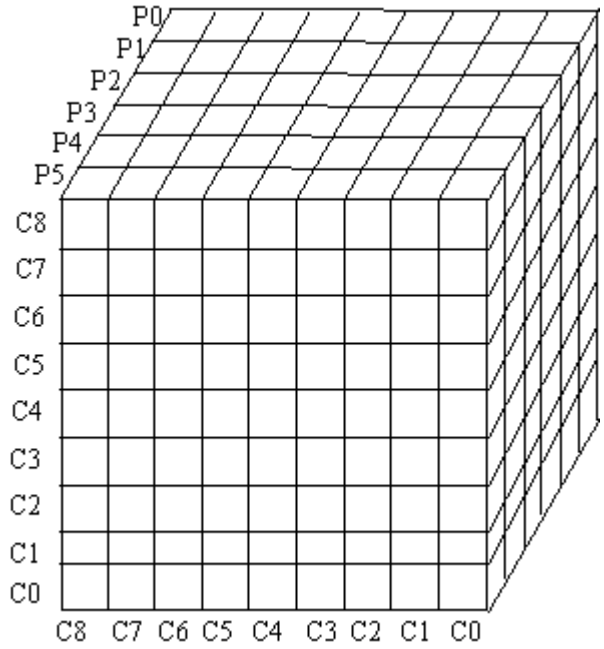
Στο [12] περιγράφεται μια εναλλακτική πρόταση, για την διαχείριση RDF Σχημάτων σε P2P συστήματα. Το σύστημα αυτό θα συγκριθεί με το σύστημά μας, καθώς και με την περίπτωση του αδόμητου συστήματος, για αυτό παρουσιάζουμε στη συνέχεια, μια αναλυτική περιγραφή του.

Η εργασία αυτή, προτείνει έναν τρόπο για αποδοτική διαχείριση RDF σχημάτων σε ένα σύστημα P2P, στο οποίο οι κόμβοι έχουν δομηθεί σε δίκτυο Chord [4]. Για να ενταχθεί ένας κόμβος στο παραπάνω σύστημα, θα πρέπει να αποθηκεύσει στο δίκτυο, το σχήμα της RDF βάσης του και να πάρει μια θέση στο Chord. Αν ένας κόμβος (δεν απαιτείται να είναι απαραίτητα στο σύστημα, αρκεί μόνο να συνδεθεί με ένα κόμβο που βρίσκεται ήδη στο σύστημα) εκδώσει ένα ερώτημα (το οποίο θα βρίσκεται σε μορφή RDF Σχήματος), τότε η εν λόγω εργασία παρέχει μια διαδικασία αναζήτησης, που εκμεταλλεύεται τη διαδικασία αναζήτησης του Chord και την οποία θα περιγράψουμε παρακάτω.

4.5.1. Περιγραφή τοπολογίας και δομικών στοιχείων

Το σύστημα αποτελείται από ένα σύνολο κόμβων, δομημένων σε ένα δίκτυο Chord. Κάθε κόμβος που εισάγεται στο σύστημα δημοσιοποιεί το σχήμα της βάσης του, μέσω της γλώσσας RVL [20]. Για να το κάνει αυτό, θα πρέπει να παράξει έναν αριθμό, ο οποίος θα χρησιμεύσει ως κλειδί, για την αναζήτηση στο chord, του κόμβου successor, στον οποίο θα αποθηκευθεί το σχήμα. Για το λόγο αυτό, κάθε κόμβος αποθηκεύει έναν εικονικό κύβο, ο οποίος ονομάζεται AdjSub Cube. Ο κύβος αυτός προκύπτει από το καθολικό σχήμα, από το οποίο προκύπτουν τα διάφορα σχήματα των κόμβων, ως εξής. Οι κλάσεις του καθολικού σχήματος (όπως και οι ιδιότητες του) αποτελούν μια ιεραρχία κόμβων-κλάσεων, οπότε μπορούν να αριθμηθούν με μια post-order αρίθμηση (κάτι ανάλογο με το δικό μας σύστημα, στο οποίο έχουμε αριθμήσει τις κλάσεις του καθολικού σχήματος με μια pre-order αρίθμηση (βλ. Κεφάλαιο 2)). Κάθε τέτοιος κόμβος-κλάση (ή αντίστοιχα, κάθε κόμβος-ιδιότητα), κρατάει το δικό του αριθμό και τον αριθμό του τελευταίου του απογόνου. Έτσι, δημιουργείται ένα διάστημα τιμών [start, end], με το οποίο μπορούμε να συγκρίνουμε ένα κόμβο-κλάση u με έναν άλλο κόμβο-κλάση v, για να δούμε πια είναι η σχέση μεταξύ τους. Για παράδειγμα, ο u είναι απόγονος του v αν και μόνο αν start(u) ≥ start(v) && end(u) < end(v). Επίσης, με βάση την παραπάνω αρίθμηση δημιουργείται και ο AdjSub Cube. Οι άξονες x και y του κύβου, αριθμούνται με βάση την αρίθμηση των κλάσεων και ο άξονας z, με βάση την αρίθμηση των ιδιοτήτων, όπως φαίνεται στο Σχ. 4.2. Ο κύβος αυτός, χωρίζεται σε |C|*|C|*|P| κελιά, όπου |C| ο συνολικός αριθμός των κλάσεων του καθολικού σχήματος και |P| ο συνολικός αριθμός των ιδιοτήτων. Με αυτόν τον τρόπο, κάθε κελί συμβολίζει μια ενδεχόμενη σχέση μεταξύ της κλάσης του άξονα x (domain) και της κλάσης του άξονα y (range), μέσω της ιδιότητας του άξονα z (property). Με άλλα λόγια, ο κύβος αυτός κρατάει όλες τις ενδεχόμενες τριάδες, που μπορεί να έχουν τα σχήματα των κόμβων. Ένας κόμβος μπορεί με αυτόν τον κύβο, να αναπαραστήσει το σχήμα της βάσης του, αν για κάθε τριάδα (domain→property→range) του σχήματος, συμπληρώναμε με 1 το αντίστοιχο κελί του AdjSub Cube και με 0 όλα τα υπόλοιπα κελιά, τα οποία δεν αντιστοιχούν σε κάποια τριάδα του σχήματος. Επομένως, κάθε κόμβος κρατάει ένα τέτοιο συμπληρωμένο AdjSub Cube, για κάθε σχήμα του (καθώς μπορεί να δημοσιεύσει περισσότερα από ένα σχήματα). Με βάση τον συμπληρωμένο

AdjSub Cube, προκύπτει ένας μοναδικός αριθμός για κάθε σχήμα ως εξής. Για κάθε τριάδα του σχήματος μπορούμε να βρούμε έναν μοναδικό αριθμό μέσω της συνάρτη-



Σχήμα 4.2 Ο AdjSub Cube με |C| = 9 και |P| = 6

σης $f = k*|C|^2 + i*|C| + j$, όπου k = |P| - end(property), i = |C| - end(domain) και j = |C| - end(range). Στη συνέχεια, βρίσκουμε τον μοναδικό αριθμό N του σχήματος, μέσω της σχέσης $N = \Sigma_{l=0\ldots n-1} 2^{fl}$, για όλες τις τριάδες f_l (n το σύνολο των τριάδων). Επειδή ο αριθμός που προκύπτει είναι υπερβολικά μεγάλος, μπορούμε να τον αναπαραστήσουμε, ως το σύνολο των μοναδικών αριθμών της κάθε τριάδας του σχήματος, δηλαδή ως εξής: $N = \{f_0, f_1, \ldots, f_n\}$. Το κλειδί ενός σχήματος μπορεί να προκύψει από αυτόν τον μοναδικό αριθμό, από τη φόρμα $key = (f_0*f_1* \ldots *f_n) \bmod 2^m$ (διατηρώντας έτσι και τη διάταξη των σχημάτων, με βάση τον αριθμό N του κάθε σχήματος, κάτι που είναι απαραίτητο για την αναζήτηση των ερωτημάτων, όπως θα δούμε στη συνέχεια), όπου 2^m οι συνολικές θέσεις του chord. Επίσης, οι κόμβοι που συμμετέχουν στο σύστημα, είναι δομημένοι στον δακτύλιο του chord.

4.5.2. Εισαγωγή κόμβου

Όταν ένας κόμβος θέλει να εισαχθεί στο σύστημα, πρέπει να δημοσιεύσει το σχήμα της βάσης του, όπως είπαμε και πιο πάνω. Για να το κάνει αυτό, βρίσκει το κλειδί, με βάση τον μοναδικό αριθμό του σχήματος και με βάση αυτό το κλειδί αποθηκεύει το σχήμα στον κόμβο (πάνω στο chord), ο οποίος είναι ο successor του συγκεκριμένου κλειδιού. Ένας κόμβος, όμως, μπορεί να απαντήσει και οποιοδήποτε «κομμάτι» τού σχήματός του, το οποίο υπάγει οριζόντια. Κατά συνέπεια, ο κόμβος που θέλει να εισαχθεί στο σύστημα, πρέπει να εκδώσει μια σειρά από αιτήσεις lookup(subschema) στο chord, ώστε να εντοπίσει τους successor κόμβους κάθε «οριζόντιου» υποσχήματός του. Στη συνέχεια, εκδίδει ένα αίτημα store(subshema, IP) του chord, σε κάθε έναν από αυτούς τους successors, ώστε να τους ενημερώσει για τη δυνατότητά του, να μπορεί να απαντάει στο συγκεκριμένο «κομμάτι» (subschema) του σχήματος του. Πρέπει να τονιστεί, ότι ο κόμβος πρέπει να κάνει τη παραπάνω διαδικασία, μόνο για τα υποσχήματά του, τα οποία υπάγει οριζόντια και όχι για τα υποσχήματά του τα οποία υπάγει κάθετα (βλ. παράγραφο 4.1 του [12], με τον ορισμό της οριζόντιας και κάθετης υπαγωγής, όπως έχει γίνει στην παρούσα διατριβή). Στη συνέχεια, ο κόμβος που θέλει να εισαχθεί, παίρνει τη θέση του στο chord, με τους μηχανισμούς που αναλύονται στο [4]. Επομένως, για την εισαγωγή ενός κόμβου στο σύστημα, απαιτούνται K*O(logn) hops (και μηνύματα), αν K είναι ο αριθμός των υποσχημάτων που πρέπει να εισάγει ο κόμβος και n είναι ο συνολικός αριθμός των κόμβων. Αυτό συμβαίνει, γιατί απαιτούνται O(logn) hops (και μηνύματα) για κάθε υποσχήμα που πρέπει να δημοσιευθεί, ώστε να βρούμε τον successor του κάθε υποσχήματος. Επίσης, απαιτούνται O(logn) hops (και μηνύματα) για να εισαχθεί ο κόμβος στο chord. Άρα συνολικά, απαιτούνται K*O(logn) + O(logn) = K*O(logn) hops (και μηνύματα).

4.5.3. Διαγραφή κόμβου

Όταν ένας κόμβος θέλει να διαγραφεί από το σύστημα, θα πρέπει να ενημερώσει όλους τους κόμβους, οι οποίοι έχουν κάποια αναφορά σε αυτόν (δηλαδή, οποίοι κόμβοι περιλαμβάνουν κάποιο από τα υποσχήματά του), ότι δεν θα είναι πλέον στο σύστημα (άρα δεν θα είναι και το σχήμα του διαθέσιμο), ώστε οι κόμβοι αυτοί, να σβήσουν τις σχετικές αναφορές. Στη συνέχεια, το πρωτόκολλο του chord

αναλαμβάνει την υπόλοιπη διαδικασία διαγραφής (ενημέρωση finger tables, stabilize κλπ.) [4]. Επομένως, για την διαγραφή ενός κόμβου στο σύστημα, απαιτούνται K*O(logn) hops (και μηνύματα), αν K είναι ο αριθμός των (οριζόντιων) υποσχημάτων του σχήματος του κόμβου και n είναι ο συνολικός αριθμός των κόμβων. Αυτό συμβαίνει, γιατί απαιτούνται O(logn) hops (και μηνύματα) για κάθε ένα από τα K υποσχήματα που πρέπει να δημοσιευθούν, ώστε να βρούμε τον successor του κάθε υποσχήματος. Επίσης, απαιτούνται O(logn) hops (και μηνύματα) για να διαγραφεί ο κόμβος από το chord (ανανεώσεις των fingertables των υπόλοιπων κόμβων). Άρα συνολικά, απαιτούνται K*O(logn) + O(logn) = K*O(logn) hops (και μηνύματα).

4.5.4. Αναζήτηση ερωτήματος

Ένα ερώτημα, στο σύστημα που περιγράφουμε, μπορεί να γίνει, μέσω της γλώσσας ερωτημάτων RQL [21] Δεδομένου ενός ερωτήματος, η διαδικασία αναζήτησης θα πρέπει να εντοπίσει όλους τους κόμβους, των οποίων το σχήμα υπάγεται κάθετα (με τον ορισμό της κάθετης υπαγωγής, όπως τον έχουμε δώσει στην παρούσα διατριβή) από το ερώτημα. Αρχικά, εξάγεται το σχήμα του ερωτήματος, το οποίο ονομάζεται strict view, και εντοπίζεται ο κόμβος, που ενδεχομένως μπορεί να απαντήσει σε αυτό το σχήμα, με τη λειτουργία αναζήτησης lookup(strict view) του chord. Ο κόμβος, που ενδεχομένως μπορεί να απαντήσει στο strict view, ονομάζεται αρχικός κόμβος. Στη συνέχεια, εντοπίζονται όλα τα (κάθετα) υποσχήματα του strict view, μέσω της υπηρεσίας sublookup(), που παρέχεται από το σύστημα. Η υπηρεσία αυτή εκδίδει μια σειρά από αιτήματα αναζήτησης στο chord, για να εντοπίσει αν κάποιος κόμβος μπορεί να απαντήσει στο εκάστοτε ερώτημα. Η σειρά με την οποία θα γίνουν αυτά τα ερωτήματα είναι πολύ σημαντική, γιατί πρέπει να υπάρχει η εγγύηση, ότι δεν θα υπάρχει ερώτημα το οποίο θα αναφέρεται σε κόμβο, που προηγείται στον δακτύλιο του chord. Η σειρά αυτή, μπορεί να προκύψει από τον AdjSub Cube, που περιγράψαμε προηγουμένως. Διαφορετικές περιοχές του AdjSub Cube ορίζουν διαφορετικές ακολουθίες από αιτήματα αναζήτησης, που πρέπει να εκδοθούν, προκειμένου να εντοπίσουμε κόμβους που μπορούν να απαντήσουν στο ερώτημα (δηλαδή κόμβους που περιέχουν αναφορές σε κόμβους, των οποίων τα σχήματα υπάγονται στο σχήμα του αρχικού ερωτήματος). Για να επιτευχθεί η σωστή

ακολουθία ερωτημάτων, η διαδικασία αναζήτησης ξεκινάει από τον αρχικό κόμβο και καλεί την υπηρεσία sublookup(strict view, strict view, αρχικός κόμβος) [12]. Η υπηρεσία αυτή λειτουργεί ως εξής. Ξεκινάει από το κελί του AdjSub Cube που αντιστοιχεί στην πρώτη τριάδα του strict view και αντικαθιστά πρώτα το range της τριάδας, ύστερα το domain και τέλος το property της τριάδας, με τις αμέσως επόμενες κλάσεις από το πεδίο ορισμού του range, του domain και του property, αντίστοιχα, του AdjSub Cube. Για κάθε αντικατάσταση μιας κλάσης (είτε του range είτε του domain) ή μιας ιδιότητας, η sublookup εκδίδει στο chord μια lookup(subschema), όπου subschema είναι το εκάστοτε υποσχήμα, αν αντικαταστήσουμε την συγκεκριμένη τριάδα του σχήματος του εκάστοτε ερωτήματος, με την τριάδα που προέκυψε μετά την αντικατάσταση της κλάσης ή της ιδιότητας. Για κάθε αντικατάσταση είτε του range, είτε του domain, είτε του property για κάθε τριάδα του σχήματος του εκάστοτε ερωτήματος, πρέπει να αντικατασταθούν, αναδρομικά, τα range, domain και property όλων των τριάδων του σχήματος του ερωτήματος. Με αυτόν τον τρόπο, η διαδικασία εγγυάται ότι κάθε υποσχήμα που προκύπτει από μια αντικατάσταση, έχει το αμέσως μεγαλύτερο κλειδί από το προηγούμενο υποσχήμα. Για να εγγυηθεί η διαδικασία, ότι κανένα υποσχήμα του ερωτήματος δεν θα ελεγχθεί περισσότερες από μια φορές, ελέγχει αν το εκάστοτε domain, range ή property έχει ήδη αντικατασταθεί. Η όλη διαδικασία τελειώνει, όταν όλες οι τριάδες του strict view αντικατασταθούν. Απαιτεί O(logn) hops (και μηνύματα), για να εντοπιστεί ο αρχικός κόμβος που «κρατάει» το strict view και S*O(logn) hops (και μηνύματα), για να εντοπιστούν οι κόμβοι που κρατάνε τα S (κάθετα) υποσχήματα του ερωτήματος. Το S αντιπροσωπεύει όλα τα δυνατά υποσχήματα του καθολικού σχήματος, που είναι κάθετα υποσχήματα του ερωτήματος. Όπου n είναι το σύνολο των κόμβων. Επομένως, συνολικά απαιτούνται S*O(logn) hops (και μηνύματα). Στο [12] υπάρχει ένας αλγόριθμος, ο οποίος υλοποιεί την υπηρεσία sublookup.

4.5.5. Υλοποίηση του συστήματος

Για την υλοποίηση του συστήματος δημιουργήσαμε μια κλάση ChordNode, η οποία περιγράφει έναν κόμβο του συστήματος. Οι διαδικασίες εισαγωγής/διαγραφής κόμβου και αναζήτησης ερωτήματος υλοποιήθηκαν με τους αλγορίθμους που περιγράφονται στο [12] και στο [4]. Πρέπει να τονίσουμε κι εδώ, όπως και στο δικό

μας σύστημα, ότι για τη δημιουργία του δακτυλίου του chord στο σύστημα [12], χρησιμοποιήσαμε τους αλγορίθμους που αναφέρονται στο [4], οι οποίοι δίνουν $O(\log^2 n)$ πολυπλοκότητα, σε hops και πλήθος μηνυμάτων (όπου n ο αριθμός των κόμβων του δακτυλίου του chord), τόσο για την εισαγωγή κόμβου, όσο και για τη διαγραφή.

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ

5.1. Εισαγωγή

Τα πειράματα που θα περιγράψουμε στη συνέχεια, πραγματοποιήθηκαν σε μηχάνημα AMD Athlon 3800+, με μέγεθος μνήμης 1GB και σε λειτουργικό σύστημα Windows XP. Για κάθε γραφική παράσταση τρέξαμε το αντίστοιχο πείραμα 10 φορές και πήραμε τα αποτελέσματα που παρουσιάζονται στις επόμενες παραγράφους. Στον πίνακα 5.1 φαίνονται οι παράμετροι, οι οποίοι επηρεάζουν τα τρία προς σύγκριση, συστήματα.

Για τα πειράματα που θα παρουσιάσουμε στη συνέχεια, πρέπει να τονίσουμε, ότι η υπόθεση που κάνουμε σχετικά με τον αριθμό των σχημάτων και τον αριθμό των κόμβων είναι ότι και στα 3 συστήματα, κάθε κόμβος εκδίδει μόνο ένα σχήμα, άρα η σχέση τους είναι 1-1. Άρα, οι χρόνοι για την εισαγωγή των σχημάτων, στο σύστημα μας, συμπίπτουν με τους χρόνους εισαγωγής ενός κόμβου.

Επίσης, κατά την υλοποίηση, τόσο του δικού μας δακτυλίου, όσο και του δακτυλίου του chord (που χρειάζεται για την υλοποίηση του [12]), χρησιμοποιήθηκαν οι αλγόριθμοι που παρέχονται στο [4]. Άρα, όσον αφορά την εισαγωγή/διαγραφή ενός σχήματος από το σύστημά μας και την εισαγωγή/διαγραφή ενός κόμβου από το [12],

Πίνακας 5.1 Παράμετροι των 3 Συστημάτων

Συμβολισμός	*Περιγραφή*
t	Αριθμός τριάδων σχήματος
n	Αριθμός κόμβων συστήματος
s	Αριθμός υπερσχημάτων, στο σύστημά μας
ν	Αριθμός σχημάτων μεγαλύτερης ιεραρχίας του συστήματός μας
p	Αριθμός διαγραφέντων υπερσχημάτων, κατά την εισαγωγή ενός υπερσχήματος, στο σύστημά μας
c	Αριθμός παιδιών ενός διαγραφέντος υπερσχήματος, στο σύστημά μας
h	Ύψος ψηλότερης ιεραρχίας, στο σύστημά μας
S	Αριθμός όλων των δυνατών υποσχημάτων του καθολικού σχήματος, που είναι κάθετα υποσχήματα ενός ερωτήματος
K	Αριθμός όλων των δυνατών υποσχημάτων του καθολικού σχήματος, που είναι οριζόντια υποσχήματα ενός ερωτήματος

τα δύο συστήματα θα χρειάζονται $O(\log^2 n)$ hops (και μηνύματα), όπου n ο αριθμός των κόμβων, για την ενημέρωση των fingertables του δακτυλίου του συστήματός μας και του δακτυλίου του chord, αντίστοιχα (σε αντιπαράθεση με O(logn) που ήταν ο θεωρητικός χρόνος ανανέωσης των fingertables στο [4]). Έτσι, οι χρόνοι μεταβάλλονται ως εξής, σε σχέση με την ως τώρα θεωρητική ανάλυσή μας. Για το σύστημά μας:

- Εισαγωγή σχήματος. $p*O(\log^2 s) + O(h)$ hops
- Διαγραφή σχήματος. $c*O(\log^2 s)$ hops και μηνύματα

Ο χρόνος για την αναζήτηση ερωτήματος, καθώς και τα μηνύματα που απαιτούνται για την εισαγωγή/διαγραφή και για την αναζήτηση ερωτήματος, είναι ίδιοι με αυτούς που βρήκαμε στο Κεφάλαιο 3, κατά την θεωρητική ανάλυση του συστήματός μας. Όσον αφορά το σύστημα του [12], αυτό που αλλάζει είναι:

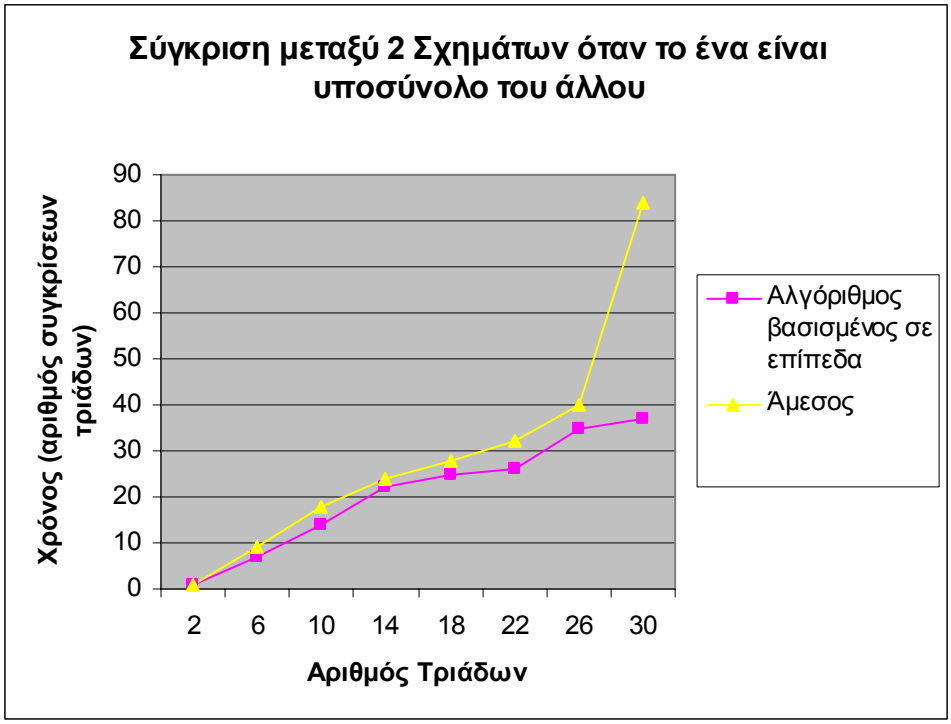
- Εισαγωγή κόμβου. $K*O(\log^2 n)$ hops και μηνύματα
- Διαγραφή κόμβου. $K*O(\log^2 n)$ hops και μηνύματα

Στη συνέχεια, θα παρουσιάσουμε τα εξής πειράματα. Στην παράγραφο 5.2, θα μελετήσουμε την απόδοση των αλγορίθμων 3.2 και 3.3, οι οποίοι χρησιμοποιούνται για να ελέγξουν ποια είναι η σχέση μεταξύ 2 σχημάτων. Στην παράγραφο 5.3, θα μελετήσουμε την απόδοση του δικού μας συστήματος, όταν μεταβάλλουμε κάποιες παραμέτρους, όπως ο αριθμός των ιεραρχιών, η επικάλυψη μεταξύ των σχημάτων κλπ. Τέλος, στην παράγραφο 5.4, θα μελετήσουμε την απόδοση του συστήματος μας, σε σχέση με το σύστημα που παρουσιάζεται στο [12] και σε σχέση με την περίπτωση του αδόμητου συστήματος.

5.2. Απόδοση Αλγορίθμων Σύγκρισης Σχημάτων

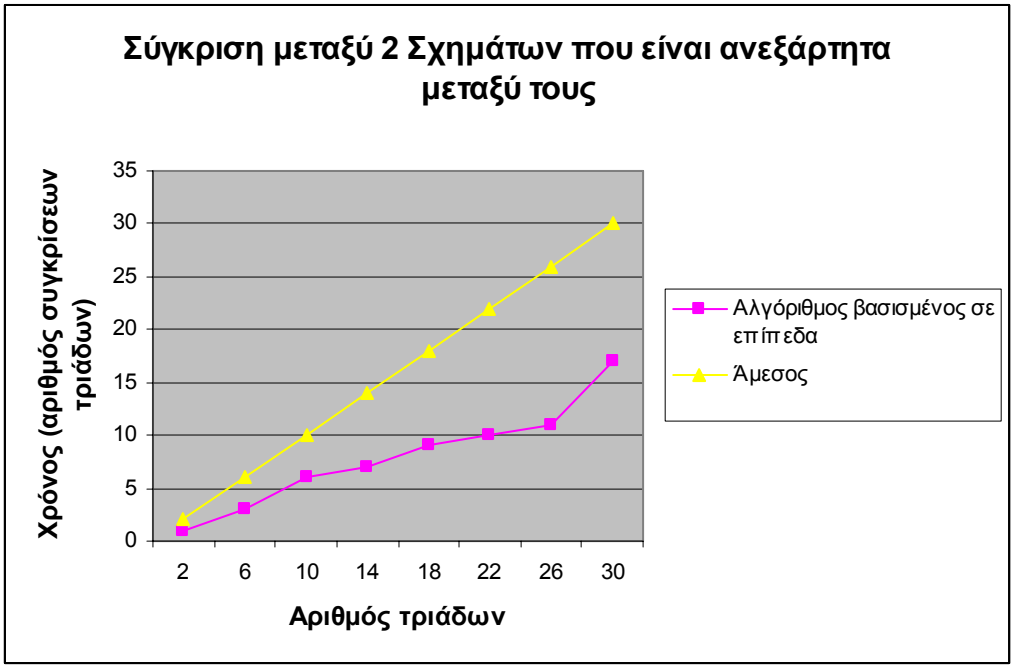
Οι αλγόριθμοι που συγκρίνουμε, είναι ο αλγόριθμος 3.2 (ο αλγόριθμος βασισμένος σε επίπεδα, που προτείνουμε εμείς) και ο αλγόριθμος 3.3 (η άμεση λύση). Η σύγκριση αφορά τον αριθμό των βημάτων, που απαιτεί ο κάθε αλγόριθμος για να περατωθεί, σε συνάρτηση με το μέγεθος των σχημάτων (σε αριθμό τριάδων). Το καθολικό σχήμα από το οποίο προκύπτουν τα σχήματα προς σύγκριση, αποτελείται από μια ιεραρχία 341 κλάσεων, με fan-out = 4 και ύψος = 5. Στα επόμενα πειράματα, στον άξονα x φαίνεται ο αριθμός των τριάδων των σχημάτων και στον άξονα y ο αριθμός των βημάτων που απαιτήθηκαν, για την περάτωση των αλγορίθμων.

Στο Σχ. 5.1, συγκρίναμε τους δύο αλγορίθμους πάνω σε σχήματα, για τα οποία ισχύει ότι το ένα είναι υποσύνολο του άλλου. Ο αλγόριθμος βασισμένος σε επίπεδα είναι ελαφρώς καλύτερος από τον άμεσο. Ωστόσο, όταν το μέγεθος των σχημάτων είναι αρκετά μεγάλο (>30 τριάδων), ο πρώτος είναι πολύ πιο αποδοτικός από τον δεύτερο. Παρατηρούμε επίσης, ότι ο άμεσος αλγόριθμος ακολουθεί την πολυπλοκότητα $O(t^2)$, ενώ ο αλγόριθμος βασισμένος σε επίπεδα ακολουθεί την πολυπλοκότητα $O(t\log^2 t)$, κάτι που επιβεβαιώνει τις θεωρητικές προβλέψεις μας.

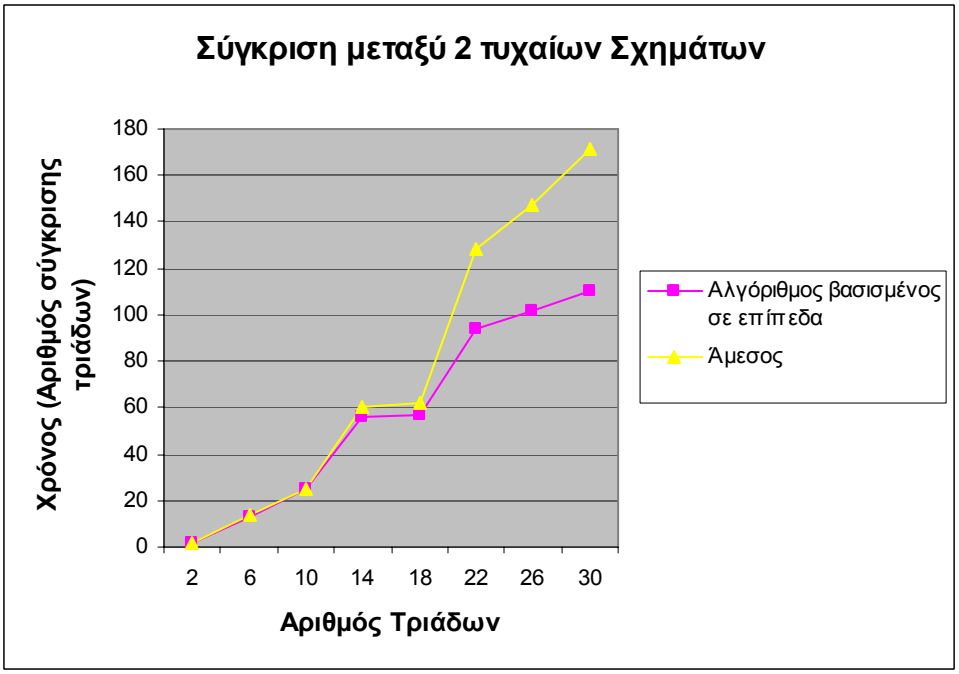


Σχήμα 5.1 Σύγκριση Αλγορίθμων όταν τα 2 Σχήματα Είναι το ένα Υποσύνολο του Άλλου

Στο Σχ. 5.2, συγκρίναμε τους δύο αλγορίθμους πάνω σε σχήματα, τα οποία είναι ανεξάρτητα μεταξύ τους. Εδώ παρατηρούμε, ότι ο αλγόριθμος βασισμένος σε επίπεδα είναι κατά πολύ καλύτερος από τον άμεσο. Ο άμεσος είναι γραμμικός στο μέγεθος των σχημάτων κι αυτό γιατί συγκρίνει την πρώτη τριάδα του πρώτου σχήματος, με όλες τις τριάδες του δευτέρου σχήματος κι αφού βρίσκει ότι καμιά τριάδα δεν υπάγεται σε κάποια άλλη (μην ξεχνάμε ότι τα σχήματα είναι ξένα μεταξύ τους), βγάζει αποτέλεσμα (δηλαδή ότι τα δύο σχήματα δεν έχουν σχέση υπαγωγής μεταξύ τους). Επίσης, παρατηρούμε, ότι οι χρόνοι στη συγκεκριμένη περίπτωση είναι πιο γρήγοροι απ' ότι στις άλλες τρεις περιπτώσεις, κάτι που περιμέναμε και που επιβεβαιώνεται από τα πειράματά μας.

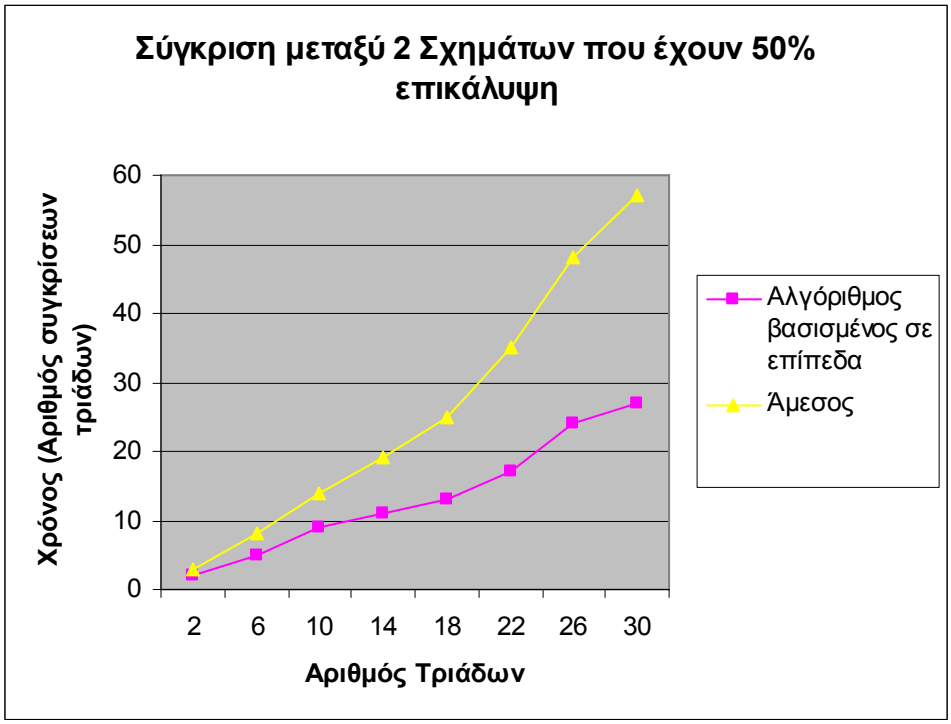


Σχήμα 5.2 Σύγκριση Αλγορίθμων όταν τα 2 Σχήματα Είναι Ανεξάρτητα μεταξύ τους



Σχήμα 5.3 Σύγκριση Αλγορίθμων όταν τα 2 Σχήματα είναι Τυχαία

Στο Σχ. 5.3, συγκρίναμε τους δύο αλγορίθμους, πάνω σε τυχαία σχήματα. Παρατηρούμε κι εδώ, ότι για μεγάλα σχήματα ο αλγόριθμος βασισμένος σε επίπεδα είναι πιο γρήγορος από τον άμεσο, εν αντιθέσει με μικρά μεγέθη σχημάτων, όπου οι δυο αλγόριθμοι είναι περίπου ίδιοι. Επίσης, παρατηρούμε ότι ακολουθούν, ο μεν άμεσος, πολυπλοκότητα $O(t^2)$, ο δε αλγόριθμος βασισμένος σε επίπεδα, πολυπλοκότητα $O(t\log^2 t)$, επιβεβαιώνοντας τις προβλέψεις μας.



Σχήμα 5.4 Σύγκριση Αλγορίθμων όταν τα 2 Σχήματα Έχουν 50% Επικάλυψη μεταξύ τους

Στο Σχ. 5.4, συγκρίναμε τους δύο αλγορίθμους, πάνω σε σχήματα τα οποία έχουν 50% επικάλυψη το ένα επί του άλλου. Δηλαδή, αν τα σχήματα περιέχουν x τριάδες, τότε θα έχουν x/2 ίδιες τριάδες ή τριάδες που έχουν κάποια σχέση υπαγωγής η μία με την άλλη. Παρατηρούμε κι εδώ, ότι ο αλγόριθμος που προτείνουμε, είναι πιο γρήγορος από τον άμεσο. Παρατηρούμε επίσης, ότι οι χρόνοι είναι πιο γρήγοροι από τις περιπτώσεις, όπου είχαμε τυχαία σχήματα ή σχήματα, όπου το ένα ήταν υποσύνολο του άλλου (σε αυτή την περίπτωση η επικάλυψη των σχημάτων είναι 100%), πράγμα που επιβεβαιώνει τις μετρήσεις μας, αφού σε αυτές τις περιπτώσεις περιμέναμε, ότι οι αλγόριθμοι θα έκαναν περισσότερο χρόνο, για να διαπιστώσουν αν

υπάρχει υπαγωγή ή όχι, απ' ότι στην περίπτωση, όπου έχουμε 50% επικάλυψη. Επίσης, οι μετρήσεις δείχνουν, ότι στη συγκεκριμένη περίπτωση, οι χρόνοι είναι πιο αργοί, σε σχέση με την περίπτωση, στην οποία τα σχήματα ήταν ανεξάρτητα μεταξύ τους (με άλλα λόγια όταν έχουν επικάλυψη 0% το ένα επί του άλλου), κάτι, που επίσης, περιμέναμε.

5.3. Απόδοση του Συστήματός μας

Στα επόμενα πειράματα θα μελετήσουμε την απόδοση του συστήματός μας, για διαφορετικές περιπτώσεις δικτύων. Για να εξάγουμε τα σχήματα του κάθε κόμβου, δημιουργήσαμε μια ιεραρχία κλάσεων και ιδιοτήτων με ύψος 7 και fan-out 4, κατά συνέπεια το πλήθος των κλάσεων (και αντίστοιχα των ιδιοτήτων) της καθολικής ιεραρχίας είναι 5462. Επίσης, τα σχήματα και τα ερωτήματα που εξάγουμε από αυτό το καθολικό σχήμα, έχουν μέγεθος 3 (εκτός και αν, σε κάποιο πείραμα, δηλώνουμε ρητά, διαφορετικό μέγεθος σχήματος ή ερωτήματος). Ποιο συγκεκριμένα θα διεξάγουμε τα εξής πειράματα:

- Μελέτη του συστήματός μας για διαφορετικές επικαλύψεις των σχημάτων του δικτύου. Πιο συγκεκριμένα, θα μελετήσουμε τις περιπτώσεις όπου τα σχήματα του δικτύου παρουσιάζουν 20% επικάλυψη μεταξύ τους, 80% επικάλυψη και τυχαία επικάλυψη. Όταν λέμε, για παράδειγμα, ότι η επικάλυψη στο δίκτυο, είναι 20%, θα εννοούμε ότι το 20% των σχημάτων του δικτύου υπάγονται σε κάποια άλλα σχήματα. Κατά συνέπεια, το 80% των σχημάτων δεν θα υπάγονται σε κανένα άλλο σχήμα (άρα θα είναι υπερσχήματα).
- Μελέτη της λειτουργίας της εισαγωγής κόμβου του συστήματός μας, όταν έχουμε διαφορετικά μεγέθη σχημάτων. Πιο συγκεκριμένα, μελετάμε τη συμπεριφορά του συστήματός μας όταν, στο σύστημα, (α) πρώτα εισέρχονται τα μεγάλα σχήματα και ύστερα τα μικρότερα (στην περίπτωση αυτή, περιμένουμε ότι το σύστημά μας θα αποδίδει καλύτερα από τις δύο επόμενες περιπτώσεις), (β) πρώτα εισέρχονται τα μικρά σχήματα και ύστερα τα μεγαλύτερα (άσχημη περίπτωση για το σύστημά μας) και (γ) τα σχήματα εισέρχονται με τυχαίο τρόπο.

- Μελέτη της λειτουργίας αναζήτησης ερωτήματος στο σύστημά μας. Πιο συγκεκριμένα, θα δούμε πόσο επηρεάζεται ο χρόνος αναζήτησης (και το πλήθος των μηνυμάτων που απαιτείται), από τα διαφορετικά μεγέθη ερωτήματος, σε σχέση και με το μέγεθος των σχημάτων στο σύστημά μας.

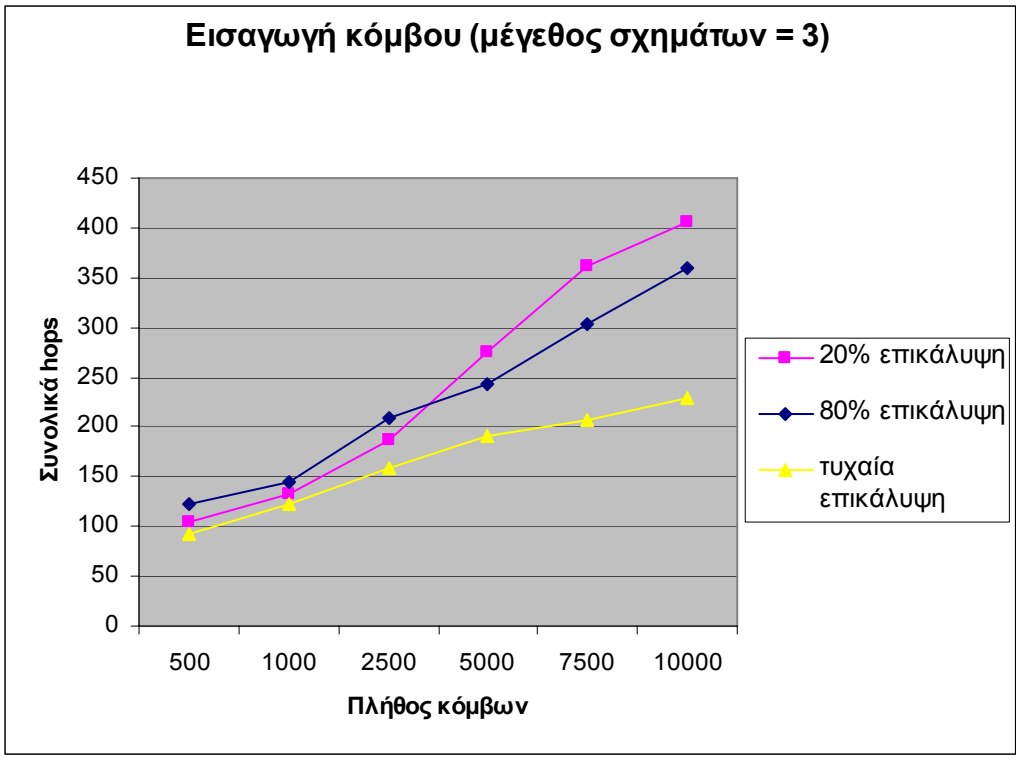
Στα παρακάτω πειράματα, μετρήσαμε τον αριθμό των hops και των μηνυμάτων σε σχέση με τον αριθμό των κόμβων του συστήματος. Πιο συγκεκριμένα, έχουμε μελετήσει 6 περιπτώσεις δικτύου (για μέγεθος δικτύου ίσο με 500, 1000, 2500, 5000, 7500 και 10000 κόμβους).

5.3.1. Μελέτη του Συστήματός μας για Διαφορετικές Επικαλύψεις Σχημάτων

Στο πρώτο μέρος των πειραμάτων, θα μελετήσουμε την επίδραση που έχουν διαφορετικά ποσοστά επικάλυψης, στο σύστημά μας. Περιμένουμε, ότι ένα μικρό ποσοστό επικάλυψης θα είναι πολύ άσχημο για το σύστημά μας, όσον αφορά το πλήθος των μηνυμάτων που απαιτούνται (και για τις τρεις λειτουργίες). Αυτό το περιμένουμε, γιατί σε αυτή την περίπτωση πολλά από τα σχήματα θα τοποθετηθούν στον δακτύλιο του συστήματός μας σαν υπερσχήματα, αφού δεν θα υπάγονται από κανένα άλλο σχήμα. Επομένως, για να λειτουργήσει το σύστημά μας, θα πρέπει να κάνει το broadcast στο δακτύλιο, σε πολλά σχήματα, κάτι που απαιτεί πολλά μηνύματα και που επηρεάζει τόσο την εισαγωγή ενός σχήματος στο δακτύλιο, όσο και την αναζήτηση ενός ερωτήματος (την διαγραφή, όχι τόσο, αφού στη διαγραφή δεν χρειάζεται να κάνουμε broadcast στο δακτύλιο).

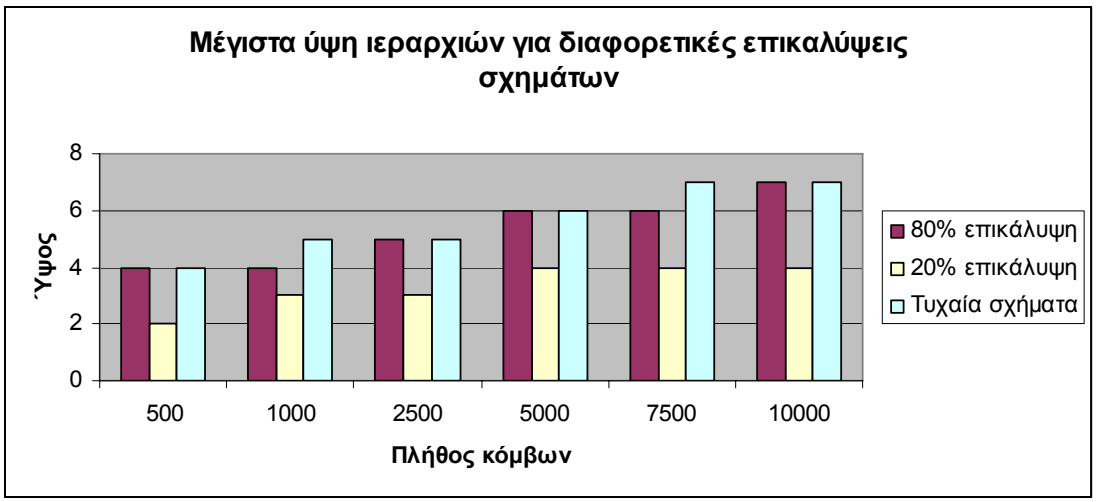
Στο Σχ. 5.5, μπορούμε να δούμε την συμπεριφορά της λειτουργίας της εισαγωγής στο σύστημά μας για 3 διαφορετικές επικαλύψεις σχημάτων (για επικάλυψη 20%, για επικάλυψη 80% και για τυχαία επικάλυψη), όσον αφορά το πλήθος των hops. Παρατηρούμε, ότι για μικρό μέγεθος δικτύου, το σύστημά μας λειτουργεί καλύτερα για επικάλυψη δικτύου 20% από ότι λειτουργεί για επικάλυψη 80%. Αντίθετα, για μεγάλα μεγέθη δικτύου, το σύστημά μας λειτουργεί καλύτερα για επικάλυψη 80%. Αυτό συμβαίνει, γιατί τα ύψη των ιεραρχιών που δημιουργούνται κατά την εισαγωγή των κόμβων είναι πιο μεγάλα για μεγαλύτερες επικαλύψεις δικτύων. Το Σχ. 5.6, μας δείχνει τα μέγιστα ύψη των ιεραρχιών σε σχέση με το μέγεθος του δικτύου και με την

επικάλυψη των σχημάτων (η πρώτη στήλη υποδεικνύει την περίπτωση, όπου έχουμε επικάλυψη 80%, η δεύτερη υποδεικνύει την περίπτωση, όπου έχουμε επικάλυψη 20% και η τρίτη την περίπτωση, όπου έχουμε τυχαία επικάλυψη). Το γεγονός ότι οι ιεραρχίες που δημιουργούνται για επικάλυψη 80% έχουν μεγαλύτερα ύψη από τις ιεραρχίες που δημιουργούνται για επικάλυψη 20%, σε συνδυασμό με το γεγονός ότι για μικρά μεγέθη δικτύου δημιουργούνται σχετικά λίγα υπερσχήματα (άρα δεν επηρεάζουν πολύ τα συνολικά hops), έχει σαν αποτέλεσμα, τα ύψη των ιεραρχιών να επηρεάζουν, εξίσου με το πλήθος των υπερσχημάτων, τα συνολικά hops που απαιτούνται. Δηλαδή, στην πολυπλοκότητα της εισαγωγής, $p*O(\log^2 s) + O(h)$, ο όρος $O(h)$ επηρεάζει το σύστημά μας, εξίσου με τον όρο $p*O(\log^2 s)$, αφού για μικρά δίκτυα το s είναι αρκετά μικρό (επίσης, το p είναι αρκετά μικρό, όπως θα δούμε και στα επόμενα πειράματα, άρα δεν επηρεάζει ουσιαστικά το σύστημά μας). Αντίθετα, για μεγάλα μεγέθη δικτύου, το πλήθος των υπερσχημάτων μεγαλώνει πάρα πολύ, με αποτέλεσμα να επηρεάζει καθοριστικά το πλήθος των hops που απαιτούνται (σε αυτή

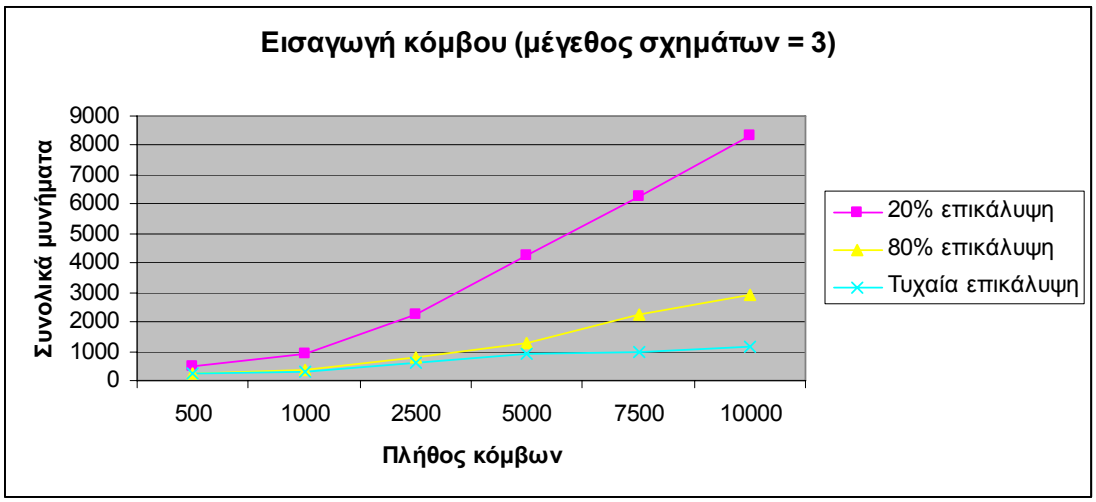


Σχήμα 5.5 Μέτρηση Εισαγωγής Κόμβου (σε Hops) στο Σύστημά μας για 3 Διαφορετικές Επικαλύψεις Σχημάτων

την περίπτωση ισχύει ότι s>>h). Αυτό που μας κάνει εντύπωση, είναι στην περίπτωση που δημιουργούμε τυχαία σχήματα, η επικάλυψη είναι μεγαλύτερη και από 80% (συνήθως η επικάλυψη πλησιάζει το 90%). Αυτό έχει ως αποτέλεσμα, το σύστημά μας για τυχαίες περιπτώσεις σχημάτων, να παρουσιάζει καλύτερη συμπεριφορά από τις άλλες δύο περιπτώσεις.

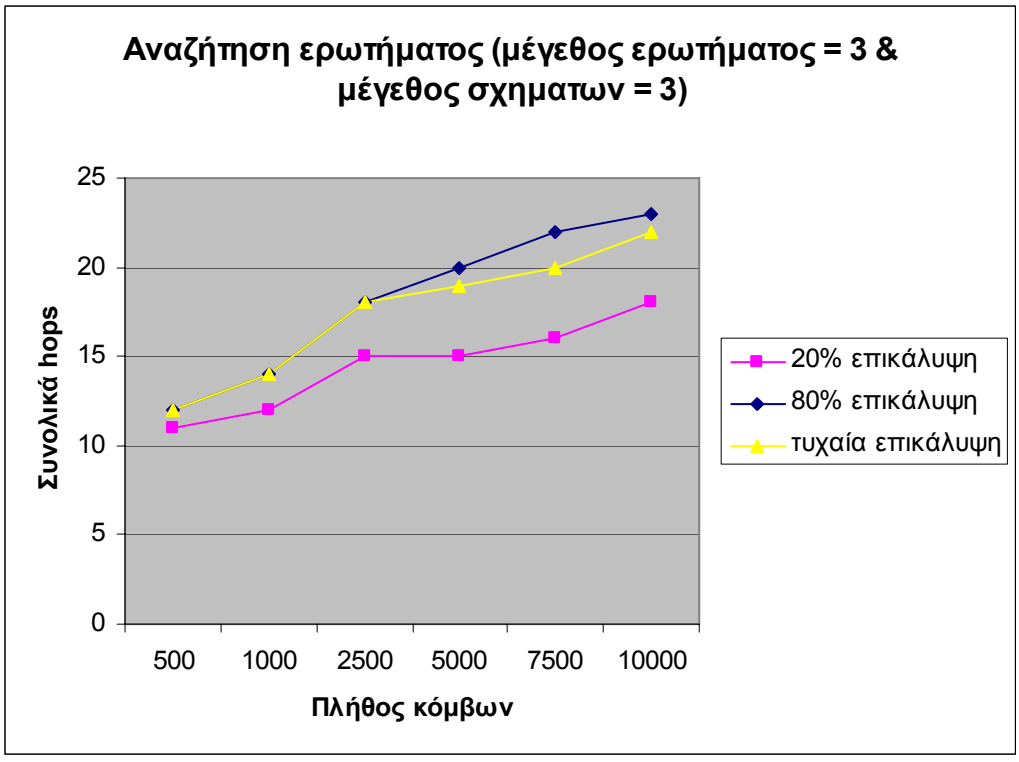


Σχήμα 5.6 Τα Ύψη των Ιεραρχιών στις Διαφορετικές Επικαλύψεις Σχημάτων στο Σύστημά μας



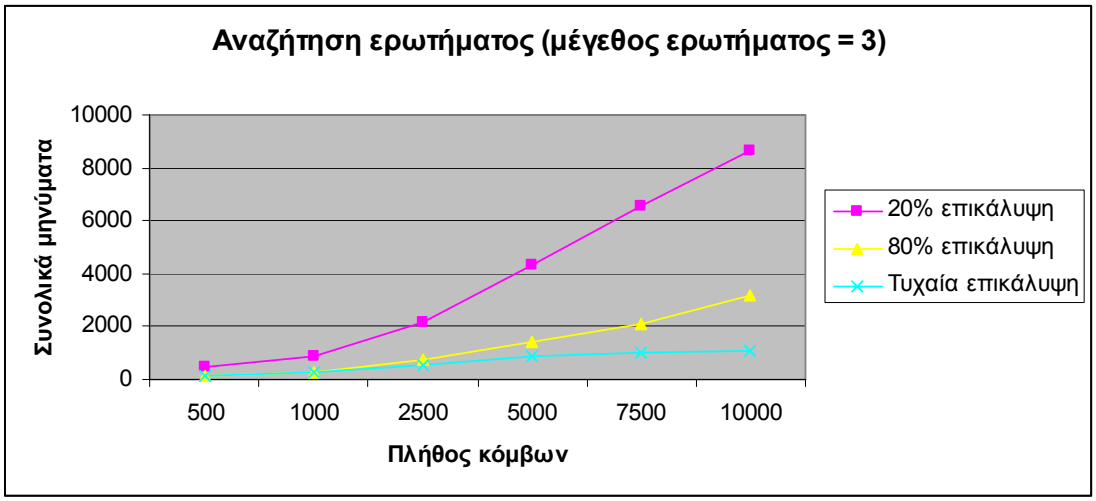
Σχήμα 5.7 Μέτρηση Εισαγωγής Κόμβου (σε Μηνύματα) στο Σύστημά μας για 3 Διαφορετικές Επικαλύψεις Σχημάτων

Στο Σχ. 5.7, μπορούμε να δούμε την συμπεριφορά της λειτουργίας της εισαγωγής στο σύστημά μας, για τις 3 επικαλύψεις σχημάτων, όσον αφορά το πλήθος των μηνυμάτων. Εδώ παρατηρούμε, ότι όσο μικρότερη είναι η επικάλυψη των σχημάτων μας, τόσο χειρότερα συμπεριφέρεται το σύστημά μας. Αυτό συμβαίνει, γιατί το broadcast που απαιτούμε να κάνει το σύστημα προκειμένου να προωθήσει ένα σχήμα σε κάθε ιεραρχία, παράγει τόσα μηνύματα, όσο και το πλήθος των υπερσχημάτων. Επειδή ο αριθμός των υπερσχημάτων αυξάνει, όσο μικρότερη είναι η επικάλυψη (για 20% επικάλυψη, το 80% των σχημάτων, τουλάχιστο, θα είναι υπερσχήματα, ενώ για 80% επικάλυψη, το 20% των σχημάτων, τουλάχιστο, θα είναι υπερσχήματα), θα αυξάνεται και το πλήθος των μηνυμάτων που απαιτούνται, για το broadcast στον δακτύλιο. Το γεγονός ότι ο αριθμός των υπερσχημάτων είναι πολύ μεγαλύτερος από τα ύψη των ιεραρχιών, κάνει το broadcast στον δακτύλιο, καθοριστικό παράγοντα για το σύστημά μας, όσον αφορά το πλήθος των μηνυμάτων. Παρατηρούμε, ότι το πλήθος των μηνυμάτων ακολουθεί την πολυπλοκότητα O(s+h) = O(s), αφού h<<s.



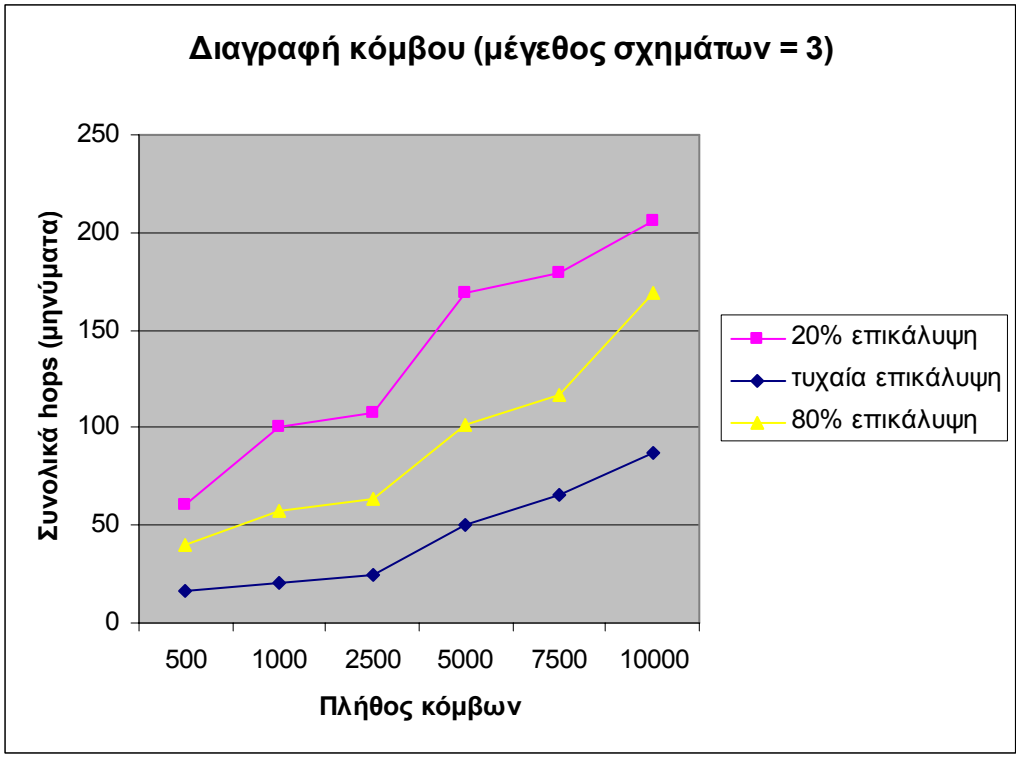
Σχήμα 5.8 Μέτρηση για την Αναζήτηση Ερωτήματος (σε Hops), για Διαφορετικές Επικαλύψεις Σχημάτων

Στο Σχ. 5.8, βλέπουμε την απόδοση του συστήματός μας, για την αναζήτηση ερωτήματος, σε διαφορετικές επικαλύψεις σχημάτων, όσον αφορά τον αριθμό των hops. Παρατηρούμε, ότι ακόμα και για μεγάλα μεγέθη δικτύων, ο αριθμός των hops που απαιτείται, για να απαντηθεί ένα ερώτημα είναι πολύ μικρός (περίπου 20 hops). Επίσης παρατηρούμε, ότι το σύστημά μας αποδίδει καλύτερα για μικρότερες επικαλύψεις δικτύων. Αυτό συμβαίνει, γιατί για να προωθηθούν τα ερωτήματα σε όλα τα υπερσχήματα απαιτείται χρόνος O(logs) hops. Άρα και για μεγέθη δικτύων ίσα με 10000 κόμβους, ο χρόνος που απαιτείται για να φτάσει ένα ερώτημα σε όλα τα υπερσχήματα είναι O(log10000) = O(14) hops (στην χειρότερη περίπτωση, όπου όλα τα σχήματα είναι υπερσχήματα). Όπως είδαμε, τα ύψη των ιεραρχιών αυξάνονται όσο μεγαλύτερη είναι η επικάλυψη, με αποτέλεσμα να φτάνουν και 7 επίπεδα (για 80% επικάλυψη και για 10000 κόμβους). Ένα τέτοιο νούμερο, όμως, σε συνδυασμό με το μικρό πλήθος των hops, που απαιτούνται για το broadcast στον δακτύλιο, έχει ως αποτέλεσμα, τα hops που απαιτούνται για να διασχίσουμε μια ιεραρχία (από την ρίζα ως το φύλλο) να επηρεάζουν καθοριστικά, το πλήθος των hops που απαιτούνται για την απάντηση ενός ερωτήματος. Μάλιστα, όσο μεγαλύτερη είναι η επικάλυψη, τόσο περισσότερο επηρεάζονται τα συνολικά hops, από το ύψος των ιεραρχιών, για αυτό και όσο μικρότερη είναι η επικάλυψη, τόσο καλύτερα αποδίδει το σύστημά μας.



Σχήμα 5.9 Μέτρηση για την Αναζήτηση Ερωτήματος (σε Μηνύματα), για Διαφορετικές Επικαλύψεις Σχημάτων

Ωστόσο, η διαφορά που παρουσιάζεται στα hops, για τις τρεις διαφορετικές επικαλύψεις, δεν είναι μεγάλη (σε σχέση και με το μέγεθος του δικτύου), οπότε μπορούμε να πούμε ότι και στις τρεις περιπτώσεις το σύστημά μας δουλεύει ικανοποιητικά.



Σχήμα 5.10 Μέτρηση για την Διαγραφή Κόμβου, για Διαφορετικές Επικαλύψεις Σχημάτων

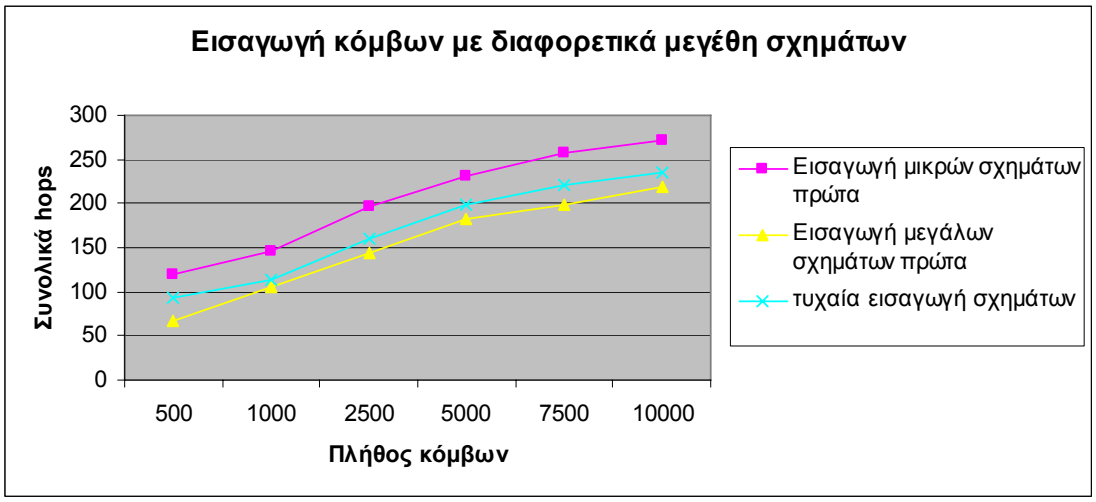
Στο Σχ. 5.9, βλέπουμε την απόδοση του συστήματός μας, για την αναζήτηση ερωτήματος, σε διαφορετικές επικαλύψεις σχημάτων, όσον αφορά τον αριθμό των μηνυμάτων. Όπως παρατηρούμε, όσο μικρότερη είναι η επικάλυψη, τόσο χειρότερα αποδίδει το σύστημά μας. Αυτό ισχύει, γιατί τώρα, το broadcast που γίνεται στο δακτύλιο, για να προωθηθούν τα ερωτήματα σε όλες τις ιεραρχίες, είναι ο καθοριστικός παράγοντας για το σύστημά μας, για τους ίδιους λόγους που είπαμε και στην περίπτωση της εισαγωγής σχήματος. Παρατηρούμε, ότι το πλήθος των μηνυμάτων ακολουθεί την πολυπλοκότητα O(s*ν), όπως περιμέναμε. Επίσης, μπορούμε να δούμε ότι το σύστημά μας, αποδίδει καλύτερα για την περίπτωση της τυχαίας επικάλυψης, αφού σε αυτήν παράγονται λιγότερα υπερσχήματα (s).

Στο Σχ. 5.10, βλέπουμε την απόδοση του συστήματός μας, για την διαγραφή κόμβου, σε διαφορετικές επικαλύψεις σχημάτων, όσον αφορά τόσο τον αριθμό των μηνυμάτων όσο και το πλήθος των hops. Παρατηρούμε, ότι όσο μεγάλη είναι η επικάλυψη, τόσο καλύτερα αποδίδει το σύστημά μας. Αυτό συμβαίνει, γιατί όσο μεγαλύτερη είναι η επικάλυψη, τόσο πιο πιθανό είναι να διαγραφεί ένας κόμβος μέσα σε μια ιεραρχία (ο κόμβος που διαγράφεται κάθε φορά επιλέγεται τυχαία). Ακόμη, όμως και να διαγραφεί ένα υπερσχήμα, αυτό θα επηρεάσει λιγότερο την περίπτωση της μεγαλύτερης επικάλυψης (απ' ότι την περίπτωση της μικρότερης επικάλυψης), αφού εκεί θα έχουμε λιγότερα υπερσχήματα, άρα η πολυπλοκότητα $c*O(\log^2 s)$, θα επηρεαστεί σε μικρότερο βαθμό (μικρότερο s). Η παράμετρος c, επίσης, δεν επηρεάζει σημαντικά το σύστημά μας για μεγάλες επικαλύψεις επειδή, παρόλο που έχουμε πολλά σχήματα σε μια ιεραρχία, από τα πειράματα προκύπτει, ότι ένας κόμβος δεν έχει περισσότερα από 3 παιδιά (για αυτή τη συγκεκριμένη περίπτωση). Επίσης, παρατηρούμε ότι ακόμα και για μεγάλα μεγέθη δικτύων, το πλήθος των μηνυμάτων (και των hops), που απαιτούνται για να γίνει η διαγραφή, παραμένει αρκετά μικρό. Αυτό συμβαίνει, γιατί στη λειτουργία της διαγραφής, δεν εμπλέκεται το broadcast του δακτυλίου. Αυτό που χρειάζεται να γίνει (μόνο στην περίπτωση της διαγραφής υπερσχήματος) είναι να ενημερωθούν τα fingertables των υπόλοιπων υπερσχημάτων, κάτι που μπορεί να γίνει σε χρόνο $O(\log^2 s)$ hops (και μηνύματα).

5.3.2. Μελέτη της Εισαγωγής Κόμβου για Διαφορετικά Μεγέθη Σχημάτων

Στο δεύτερο μέρος των πειραμάτων, θα μελετήσουμε την επίδραση που έχουν στο χρόνο εισαγωγής ενός σχήματος (και αντίστοιχα, του κόμβου που εκδίδει το κάθε σχήμα) στο σύστημά μας, η διαφορετική σειρά εισαγωγής μεγάλων και μικρών σχημάτων. Για την ακρίβεια, έχουμε δημιουργήσει 3 ομάδες από σχήματα, με διαφορετικό μέγεθος σχημάτων η καθεμία. Η πρώτη ομάδα περιλαμβάνει τα μικρότερα σχήματα, με μέγεθος 4 τριάδες. Η δεύτερη ομάδα περιλαμβάνει τα μεσαία σχήματα, με μέγεθος 12 τριάδες. Η τρίτη ομάδα περιλαμβάνει τα μεγάλα σχήματα, με μέγεθος 20 τριάδες. Μελετούμε τρεις περιπτώσεις. Στην πρώτη, εισάγουμε πρώτα τα μικρά σχήματα, ύστερα τα μεσαία και μετά τα μεγάλα. Αυτή είναι και η χειρότερη περίπτωση για το σύστημά μας, αφού περιμένουμε, ότι τα μεγαλύτερα σχήματα θα

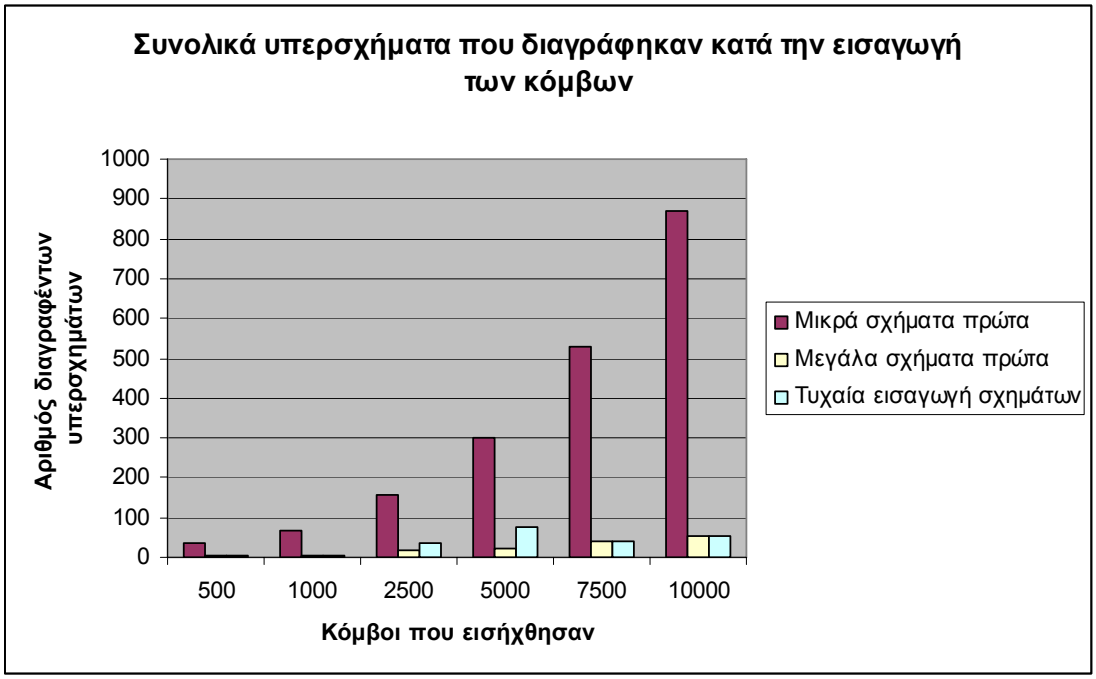
υπάγουν τα μικρότερα, με συνέπεια, να έχουμε πολλές διαγραφές σχημάτων (θα επηρεάζεται ο παράγοντας p, στην πολυπλοκότητα $p*O(\log^2 s) + O(h)$), κατά την εισαγωγή των μεγαλύτερων σχημάτων. Η καλύτερη περίπτωση για το σύστημά μας είναι όταν εισάγονται πρώτα τα μεγαλύτερα σχήματα, αφού όταν εν συνεχεία θα εισαχθούν τα μικρότερα, αυτά θα βρουν την ιεραρχία, στην οποία ανήκουν, χωρίς να απαιτηθεί καμία διαγραφή.



Σχήμα 5.11 Μέτρηση Εισαγωγής Κόμβου (σε Hops) για Διαφορετικά Μεγέθη Σχημάτων

Στο Σχ. 5.11 παρατηρούμε την συμπεριφορά του συστήματός μας στην εισαγωγή κόμβου (όσον αφορά τα συνολικά hops), όταν οι κόμβοι εκδίδουν σχήματα διαφορετικού μεγέθους. Παρατηρούμε, ότι και για τις τρεις περιπτώσεις που μελετούμε το σύστημά μας συμπεριφέρεται ικανοποιητικά. Χειρότερα συμπεριφέρεται για την περίπτωση της εισαγωγής μικρών σχημάτων πρώτα, αφού σε αυτή την περίπτωση, ο αριθμός των υπερσχημάτων που πρέπει να διαγραφούν είναι μεγαλύτερος από τις άλλες δύο περιπτώσεις. Αντίθετα, οι άλλες 2 περιπτώσεις συμπεριφέρονται σχεδόν το ίδιο ικανοποιητικά (λίγο χειρότερα συμπεριφέρεται στην τυχαία εισαγωγή σχημάτων). Τον συνολικό αριθμό των υπερσχημάτων που διαγράφονται κατά την εισαγωγή διαφορετικού αριθμού κόμβου, μπορούμε να το δούμε στο Σχ. 5.12. Στην αριστερή στήλη φαίνεται η περίπτωση στην οποία πρώτα εισάγονται τα μικρά σχήματα πρώτα, στη μεσαία στήλη φαίνεται η περίπτωση στην

οποία πρώτα εισάγονται τα μεγάλα σχήματα πρώτα και στη δεξιά στήλη φαίνεται η περίπτωση στην οποία τα σχήματα εισάγονται τυχαία.

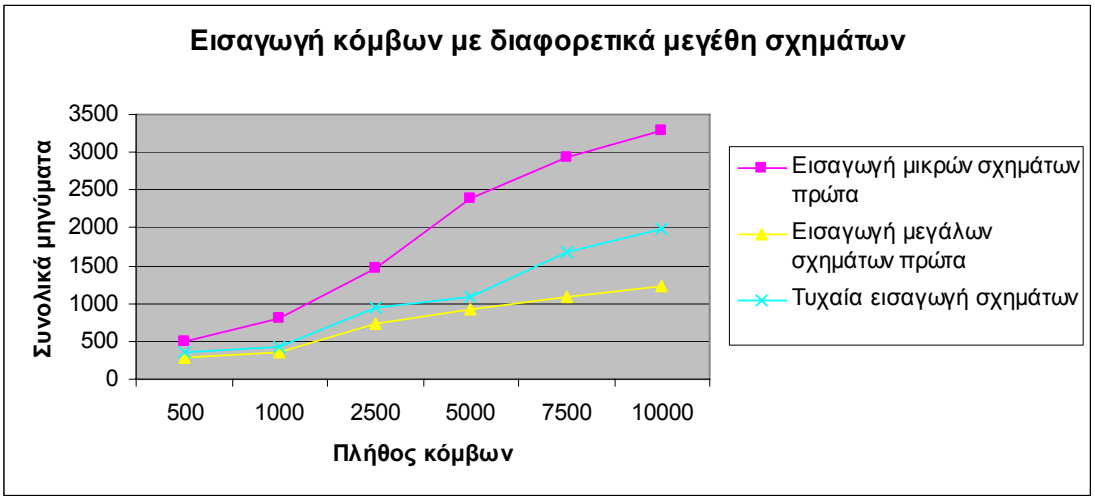


Σχήμα 5.12 Συνολικά Υπερσχήματα που Διαγράφηκαν κατά την Εισαγωγή Κόμβου για 3 Περιπτώσεις Σχημάτων

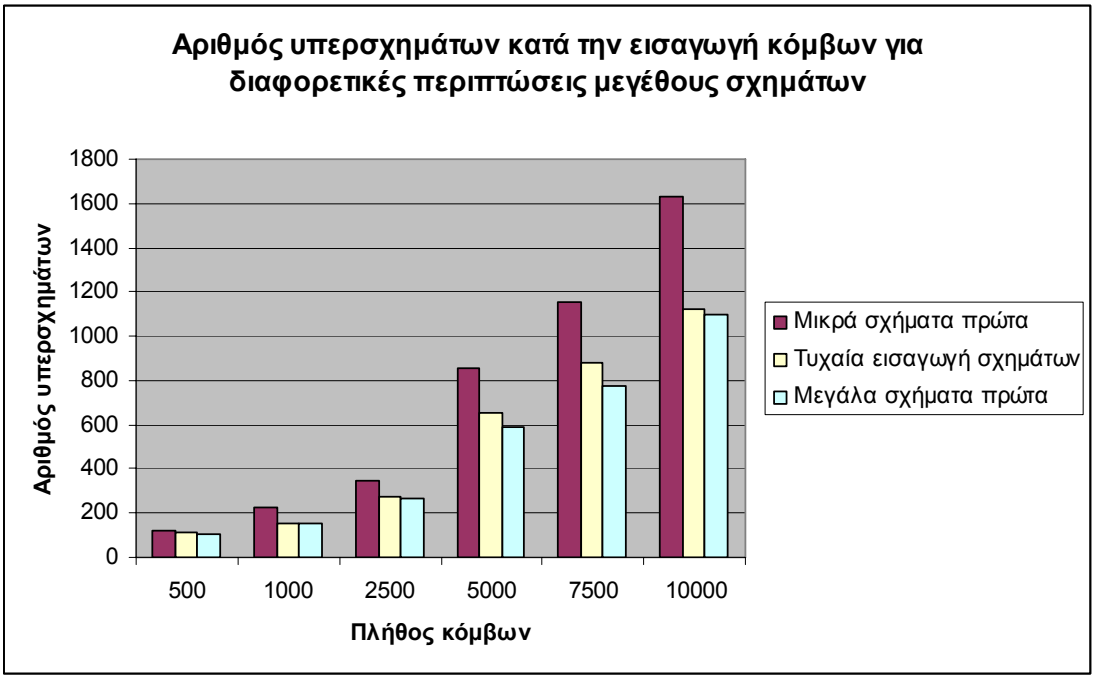
Παρατηρούμε ότι ο αριθμός των σχημάτων που διαγράφονται, όταν πρώτα εισάγονται τα μικρά σχήματα στο σύστημά μας, είναι πολύ μεγαλύτερος από τις άλλες δύο περιπτώσεις. Ωστόσο, δεν είναι αρκετά μεγάλος για να επηρεάσει, καθοριστικά, το σύστημά μας (για παράδειγμα, για τις εισαγωγές των 10000 κόμβων στο σύστημά μας, είχαμε συνολικά, ~900 διαγραφές υπερσχημάτων). Για το λόγο αυτό, τα συνολικά hops που απαιτούνται, και για τις τρεις περιπτώσεις, είναι περίπου ίδια (δηλαδή, ο παράγοντας p, στην πολυπλοκότητα $p*O(\log^2 s) + O(h)$, είναι πολύ μικρός και δεν επηρεάζει σημαντικά τα συνολικά hops).

Στο Σχ. 5.13 παρατηρούμε την συμπεριφορά του συστήματός μας στην εισαγωγή κόμβου (όσον αφορά τα συνολικά μηνύματα), όταν οι κόμβοι εκδίδουν σχήματα διαφορετικού μεγέθους. Κι εδώ, το σύστημά μας συμπεριφέρεται χειρότερα στην περίπτωση της εισαγωγής μικρών σχημάτων πρώτα. Λογικά, θα περιμέναμε να μην υπάρχει τόση μεγάλη διαφορά μεταξύ των τριών περιπτώσεων, δεδομένου ότι, (α)

στη μελέτη των hops η διαφορά μεταξύ των τριών συστημάτων ήταν μικρή και (β) η διαφορά μεταξύ των hops και των μηνυμάτων οφείλεται μόνο στο broadcast του



Σχήμα 5.13 Μέτρηση Εισαγωγής Κόμβου (σε Μηνύματα) για Διαφορετικά Μεγέθη Σχημάτων



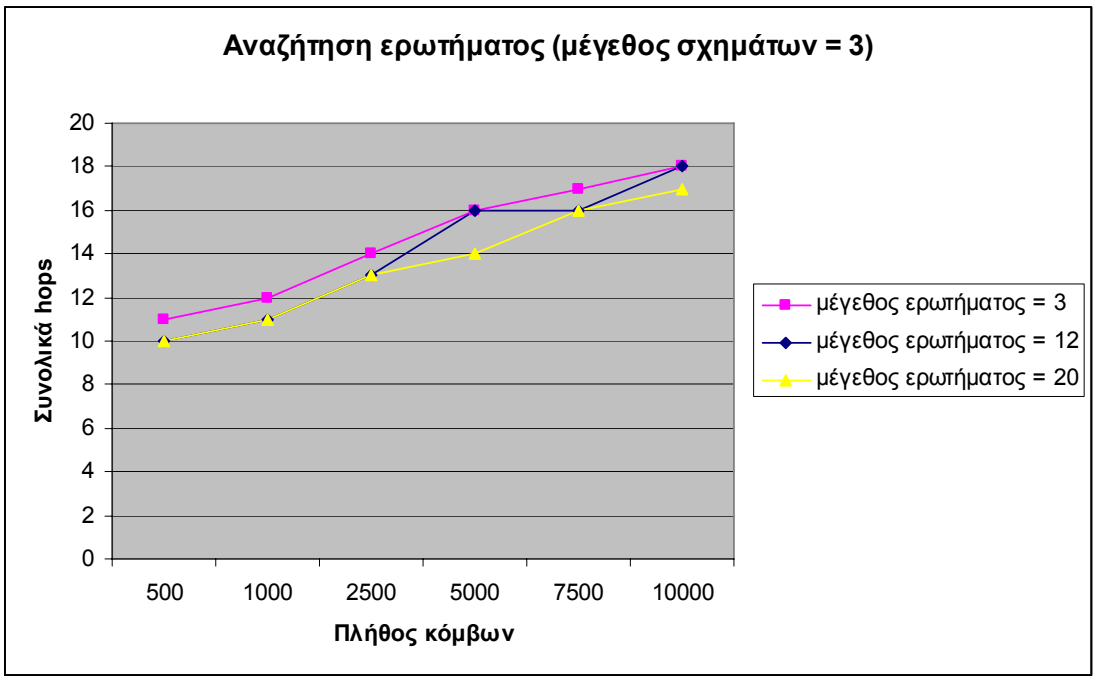
Σχήμα 5.14 Αριθμός Υπερσχημάτων που Δημιουργούνται για 3 Περιπτώσεις Εισαγωγής Σχημάτων

δακτυλίου (αφού μπορεί να γίνει με λίγα hops, αλλά με πολλά μηνύματα). Επομένως, η διαφορά που παρατηρείται μεταξύ των μηνυμάτων των 3 συστημάτων, μπορεί να εξηγηθεί μόνο, αν κατά τις τρεις περιπτώσεις δημιουργείται διαφορετικός αριθμός υπερσχημάτων. Πράγματι, τα πειράματα έδειξαν ότι στην περίπτωση που εισάγουμε πρώτα τα μικρά σχήματα, τείνουν να σχηματιστούν περισσότερα υπερσχήματα από τις άλλες 2 περιπτώσεις. Για την ακρίβεια, στο Σχ. 5.14, παρατηρούμε τον αριθμό των υπερσχημάτων που δημιουργούνται στις τρεις περιπτώσεις που μελετάμε, για κάθε μέγεθος δικτύου. Στην αριστερή στήλη φαίνεται η περίπτωση στην οποία πρώτα εισάγονται τα μικρά σχήματα, στη μεσαία στήλη φαίνεται η περίπτωση στην οποία τα σχήματα εισάγονται τυχαία και στη δεξιά στήλη φαίνεται η περίπτωση στην οποία πρώτα, εισάγονται τα μεγάλα σχήματα.

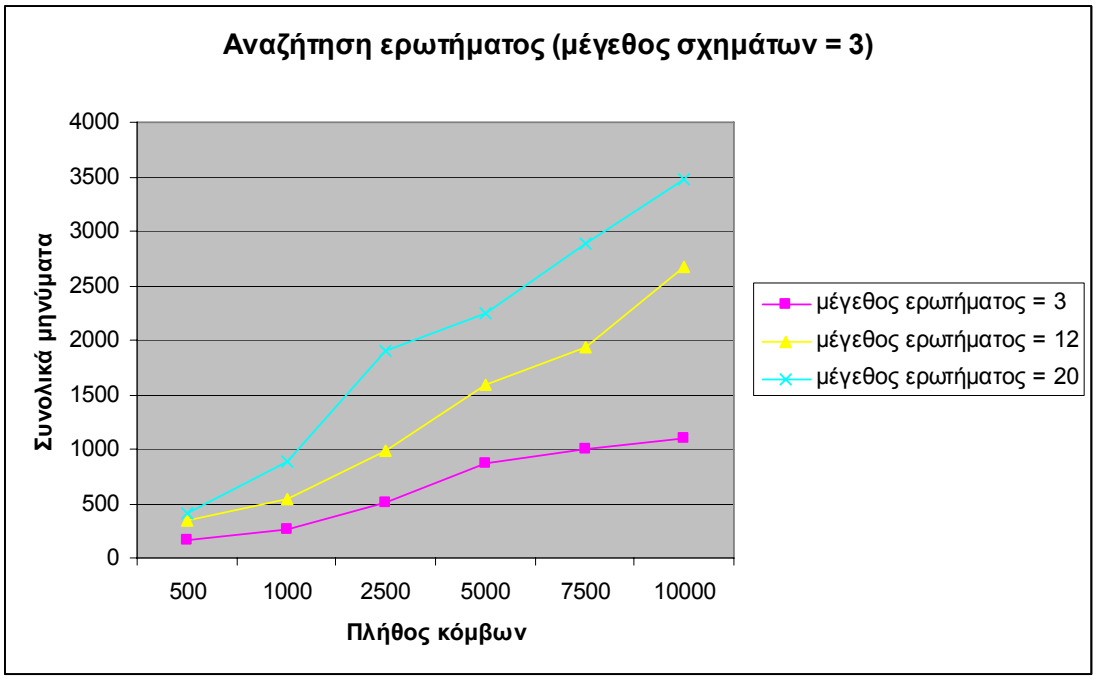
5.3.3. Μελέτη Αναζήτησης Ερωτήματος για Διαφορετικά Μεγέθη Ερωτήματος

Στο τρίτο μέρος των πειραμάτων, μελετάμε τους χρόνους αναζήτησης για διαφορετικά μεγέθη ερωτημάτων και σε δίκτυα με διαφορετικά μεγέθη σχημάτων. Για την ακρίβεια, έχουμε μελετήσει τρεις περιπτώσεις δικτύων: (α) δίκτυα με μεγέθη σχημάτων ίσα με 4, (β) δίκτυα με μεγέθη σχημάτων ίσα με 12 και (γ) δίκτυα με μεγέθη σχημάτων ίσα με 20 τριάδες. Για τις τρεις αυτές, διαφορετικές περιπτώσεις δικτύων, μελετούμε διαφορετικά μεγέθη ερωτήματος και πιο συγκεκριμένα για μεγέθη ερωτήματος 4, 12 και 20 τριάδων. Περιμένουμε, ότι το μέγεθος του ερωτήματος θα επηρεάζει μόνο τα μηνύματα που απαιτούνται από το σύστημα, για να απαντηθεί το εκάστοτε ερώτημα. Αυτό, γιατί κάθε κόμβος που μπορεί να απαντήσει ένα μέρος του ερωτήματος, θα «συνεισφέρει» κι από ένα μήνυμα. Από την άλλη, τα μεγέθη των ερωτημάτων, δεν περιμένουμε να επηρεάζουν το συνολικό αριθμό των hops, αφού αυτός επηρεάζεται μόνο από τον αριθμό των υπερσχημάτων και από το ύψος των ιεραρχιών.

Στα Σχ. 5.15 και 5.16, παρατηρούμε την απόδοση του συστήματός μας για την αναζήτηση ερωτήματος, για 3 διαφορετικά μεγέθη ερωτήματος, όταν το μέγεθος των σχημάτων είναι 3. Παρατηρούμε, ότι τα μεγέθη ερωτήματος δεν επηρεάζουν τα hops του συστήματός μας. Αυτό συμβαίνει, γιατί αφενός, το ερώτημα θα πρέπει να δρομολογηθεί, έτσι κι αλλιώς, σε όλα τα υπερσχήματα κι αφετέρου, τα ύψη των



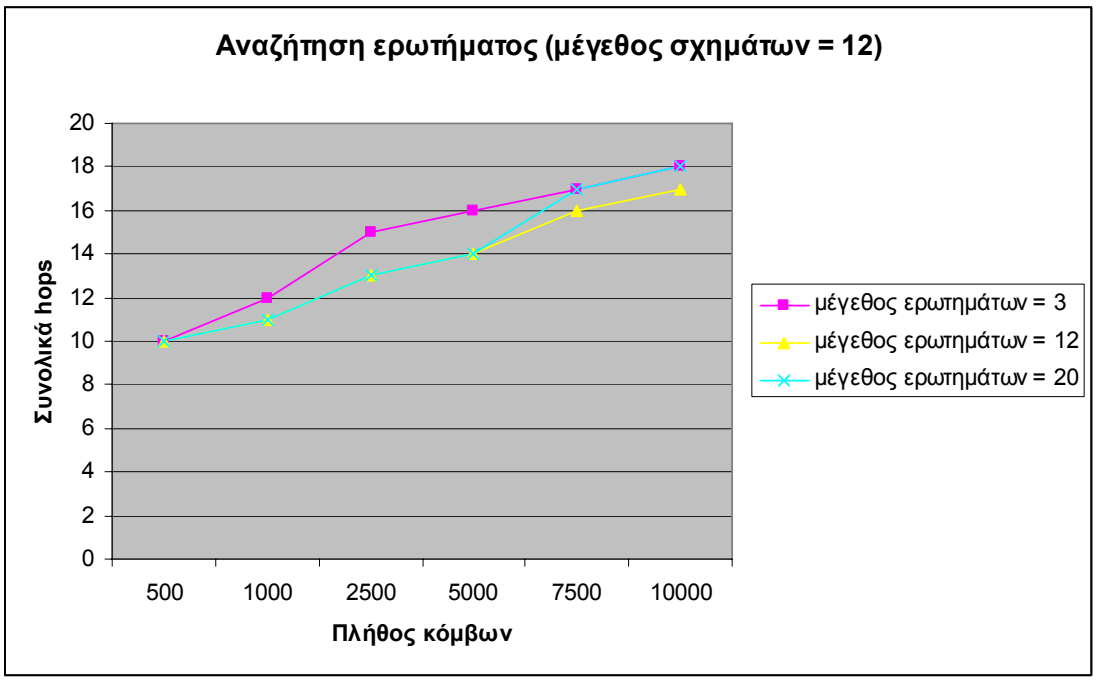
Σχήμα 5.15 Μέτρηση Αναζήτησης Ερωτήματος (σε Hops) για Μέγεθος Σχημάτων = 3



Σχήμα 5.16 Μέτρηση Αναζήτησης Ερωτήματος (σε Μηνύματα) για Μέγεθος Σχημάτων = 3

ιεραρχιών δεν εξαρτώνται από το μέγεθος των ερωτημάτων, όποτε οι χρόνοι σε hops, για όλα τα ερωτήματα θα είναι ίδιοι.

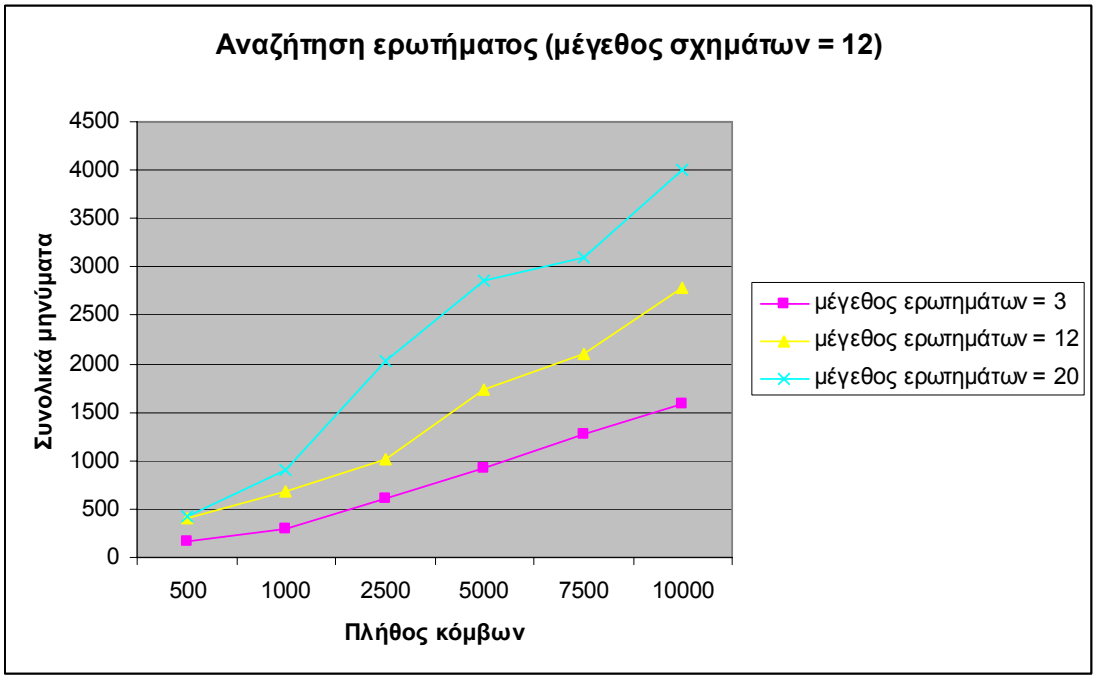
Από την άλλη, παρατηρούμε ότι τα διαφορετικά μεγέθη ερωτήματος επηρεάζουν τα συνολικά μηνύματα που απαιτούνται για να απαντηθεί ένα ερώτημα. Αυτό συμβαίνει επειδή, όσο μεγαλώνει ένα ερώτημα, τόσο περισσότερα σχήματα μπορούν να το απαντήσουν (στην πολυπλοκότητα O(s*ν), αυξάνει το ν). Πρέπει να τονίσουμε, ότι αυτό που αυξάνεται είναι ο αριθμός των σχημάτων, εντός των ιεραρχιών που μπορούν να απαντήσουν το ερώτημα (και όχι ο αριθμός των υπερσχημάτων, αφού το μήνυμα θα προωθηθεί στα υπερσχήματα με broadcast, ούτως ή άλλως).



Σχήμα 5.17 Μέτρηση Αναζήτησης Ερωτήματος (σε Hops) για Μέγεθος Σχημάτων = 12

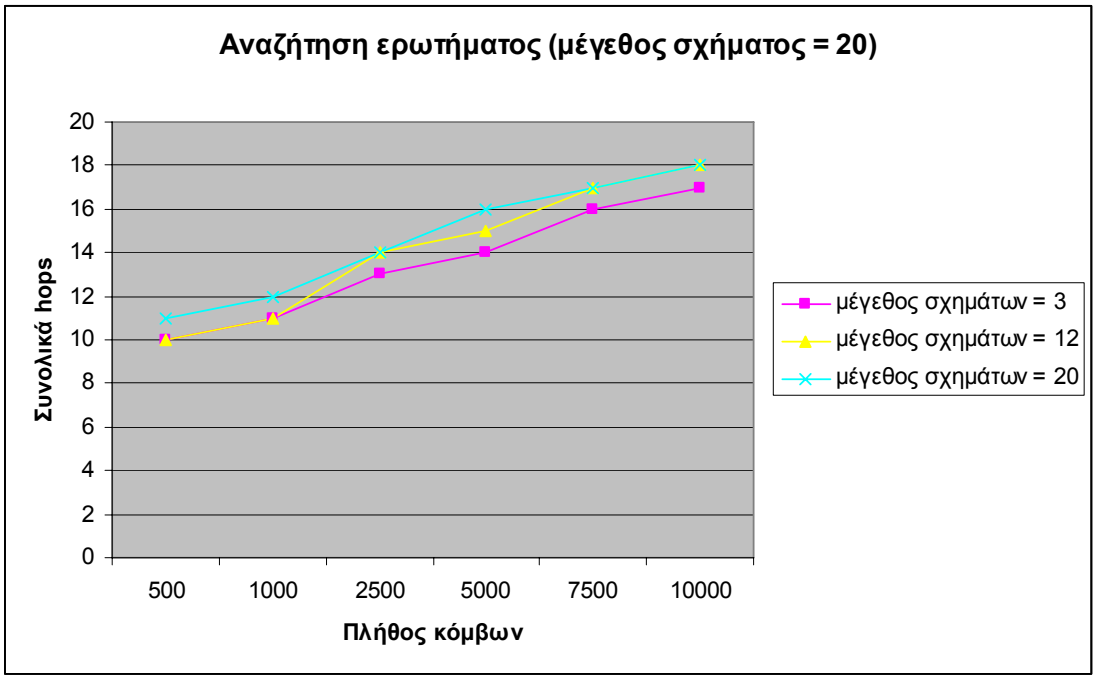
Στα Σχ. 5.17 και 5.18, παρατηρούμε την απόδοση του συστήματός μας για την αναζήτηση ερωτήματος, για 3 διαφορετικά μεγέθη ερωτήματος, όταν το μέγεθος των σχημάτων είναι 12. Παρατηρούμε κι εδώ ότι τα hops που απαιτούνται για να απαντηθεί μια ερώτηση δεν επηρεάζονται, ούτε από το μέγεθος του ερωτήματος, ούτε από το μέγεθος των σχημάτων, αφού το πλήθος των hops, που απαιτούνται για μεγέθη σχημάτων ίσα με 12 και για μεγέθη σχημάτων ίσα με 3, είναι περίπου ίδια.

Αυτό, όμως που επηρεάζεται από το μέγεθος των σχημάτων (και από το μέγεθος του ερωτήματος, όπως είπαμε για το προηγούμενο πείραμα και βλέπουμε κι εδώ), είναι ο συνολικός αριθμός μηνυμάτων που απαιτούνται, για να απαντηθεί ένα ερώτημα. Παρατηρώντας τα Σχ. 5.16 και 5.18, βλέπουμε, ότι όταν έχουμε μεγαλύτερα μεγέθη σχημάτων, τότε αυξάνεται και ο απαιτούμενος αριθμός μηνυμάτων. Αυτό είναι κάτι που περιμέναμε, αφού τα μεγαλύτερα σχήματα, ενδεχομένως να μπορούν να απαντήσουν σε περισσότερα μέρη ενός ερωτήματος.

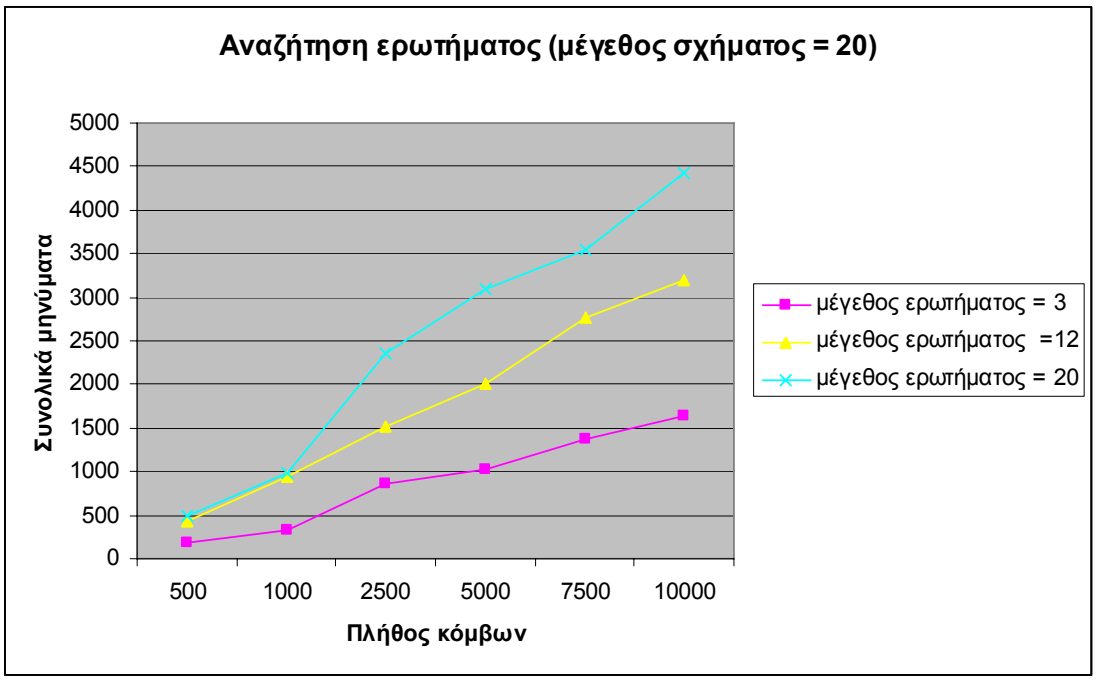


Σχήμα 5.18 Μέτρηση Αναζήτησης Ερωτήματος (σε Μηνύματα) για Μέγεθος Σχημάτων = 12

Στα Σχ. 5.19 και 5.20, παρατηρούμε την απόδοση του συστήματός μας για την αναζήτηση ερωτήματος, για 3 διαφορετικά μεγέθη ερωτήματος, όταν το μέγεθος των σχημάτων είναι 20. Παρατηρούμε κι εδώ, ότι τα hops που απαιτούνται για να απαντηθεί μια ερώτηση δεν επηρεάζονται, ούτε από το μέγεθος του ερωτήματος, ούτε από το μέγεθος των σχημάτων, όπως εξηγήσαμε και για τα προηγούμενα μεγέθη σχημάτων. Επίσης, παρατηρούμε κι εδώ, ότι όσο αυξάνεται το μέγεθος των σχημάτων και το μέγεθος των ερωτημάτων, τόσο αυξάνεται και το πλήθος των απαιτούμενων μηνυμάτων.



Σχήμα 5.19 Μέτρηση Αναζήτησης Ερωτήματος (σε Hops) για Μέγεθος Σχημάτων = 20



Σχήμα 5.20 Μέτρηση Αναζήτησης Ερωτήματος (σε Μηνύματα) για Μέγεθος Σχημάτων = 12

5.4. Σύγκριση Συστημάτων

Τα παρακάτω πειράματα συγκρίνουν το σύστημα που προτείνουμε στην παρούσα εργασία, με 2 άλλα συστήματα, στις λειτουργίες της εισαγωγής/διαγραφής κόμβου και αναζήτησης ερωτήματος. Το πρώτο από τα δύο συστήματα είναι το [12] και το δεύτερο ένα σύστημα, στο οποίο οι κόμβοι ενώνονται τυχαία μεταξύ τους, δημιουργώντας ένα αδόμητο δίκτυο.

Στα παρακάτω πειράματα χρησιμοποιήθηκαν τυχαία σχήματα με μέγεθος 12 (συνολικός αριθμός τριάδων). Κατά συνέπεια, οι ιεραρχίες που σχηματίζονται στο σύστημα μας είναι τυχαίες (τόσο το μέγεθος των ιεραρχιών, όσο και το ύψος κάθε ιεραρχίας) για κάθε πείραμα και για κάθε μέγεθος δικτύου. Τα σχήματα που εκδίδουν οι κόμβοι έχουν εξαχθεί από μια καθολική ιεραρχία 341 κλάσεων (και ιδιοτήτων) με fan-out = 4 και ύψος = 5. Παρουσιάζουμε 2 κατηγορίες πειραμάτων. Στην πρώτη, μελετάμε τον αριθμό των hops σε σχέση με το μέγεθος του δικτύου για την εισαγωγή/διαγραφή σχήματος από το σύστημα και για την αναζήτηση ερωτήματος. Σε αυτή την κατηγορία, στα πειράματα που θα παρουσιαστούν, ο άξονας y παριστάνει τον αριθμό των hops. Στη δεύτερη κατηγορία, μελετάμε τον αριθμό των μηνυμάτων σε σχέση με το μέγεθος του δικτύου για την εισαγωγή/διαγραφή σχήματος από το σύστημα και για την αναζήτηση ερωτήματος. Εδώ, στα πειράματα που θα παρουσιαστούν, ο άξονας y παριστάνει τον αριθμό των μηνυμάτων. Επίσης, και για τις δύο περιπτώσεις, ο άξονας x παριστάνει τον αριθμό των κόμβων του δικτύου (500 - 10000).

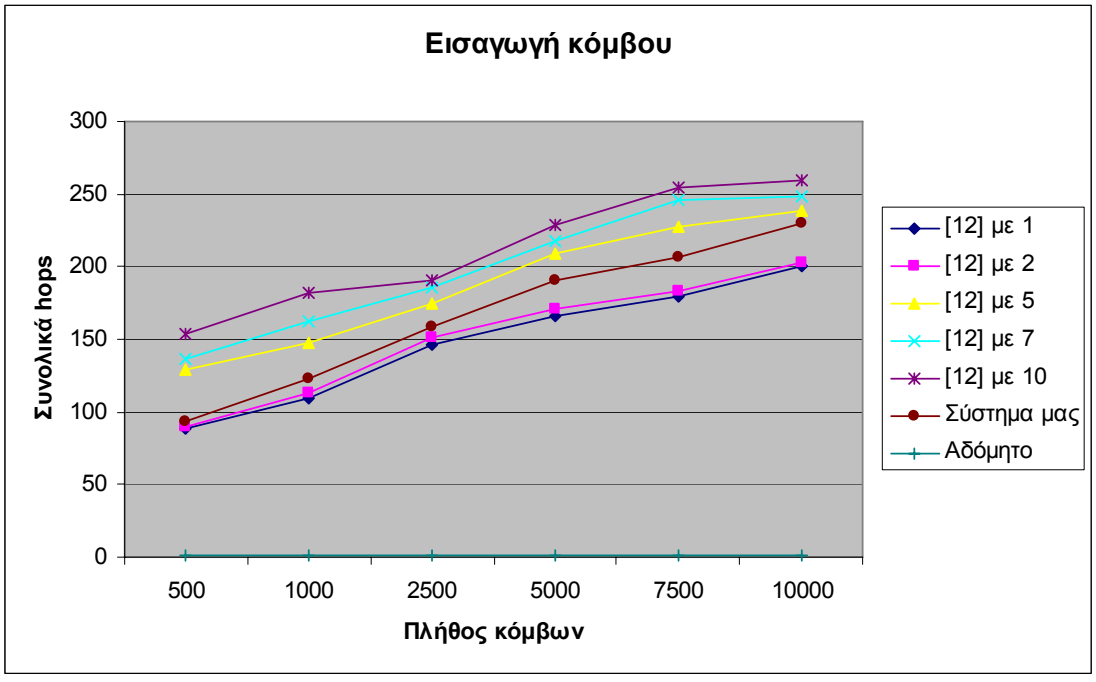
Όπως είπαμε και στην περιγραφή του συστήματος [12], η εισαγωγή/διαγραφή ενός κόμβου σε αυτό το σύστημα, εξαρτάται από των αριθμό των οριζόντιων υποσχημάτων του σχήματος του συγκεκριμένου κόμβου. Για το λόγο αυτό, στα πειράματα που θα δείξουμε στη συνέχεια (σε όσα αφορούν την εισαγωγή/διαγραφή κόμβου), θα συγκρίνουμε το σύστημα μας, την παραλλαγή του αδόμητου συστήματος και 5 περιπτώσεις για το σύστημα [12]. Πιο συγκεκριμένα, θα συγκρίνουμε τις

περιπτώσεις, στις οποίες τα σχήματα των κόμβων (που εισάγονται/διαγράφονται από το σύστημα [12]) έχουν 1, 2, 5, 7 και 10 οριζόντια υποσχήματα. Για να θυμίσουμε, ένα σχήμα S_1 είναι οριζόντιο υποσχήμα ενός σχήματος S_2, αν και μόνο αν κάθε statement του S_1 ανήκει και στο S_2. Επίσης, είπαμε ότι η αναζήτηση ενός ερωτήματος στο [12], εξαρτάται από τον αριθμό των κάθετων υποσχημάτων, του σχήματος ενός συγκεκριμένου ερωτήματος. Για να θυμίσουμε, ένα σχήμα S_1 είναι κάθετο υποσχήμα ενός σχήματος S_2, αν και μόνο αν για κάθε statement (τριάδα) st_1 του S_1 υπάρχει ένα statement st_2 του S_2, ώστε το st_1 να υπάγεται στο st_2. Ο αριθμός των κάθετων υποσχημάτων εξαρτάται, έμμεσα, από το μέγεθος του ερωτήματος (από τον αριθμό των τριάδων του σχήματός του). Για το λόγο αυτό, όσον αφορά την αναζήτηση ερωτήματος, συγκρίνουμε το σύστημά μας, την παραλλαγή του αδόμητου συστήματος και το [12], για 4 διαφορετικά μεγέθη ερωτήματος (3, 8 και 12 τριάδες). Πρέπει να τονίσουμε, ότι το δικό μας σύστημα, όπως και η παραλλαγή του αδόμητου συστήματος δεν εξαρτώνται, άμεσα, από το μέγεθος των σχημάτων. Από το μέγεθος των σχημάτων εξαρτάται μόνο ο χρόνος του αλγορίθμου σύγκρισης σχημάτων, τον οποίο μελετήσαμε στην παράγραφο 3.4.

5.4.1. Εισαγωγή Κόμβου

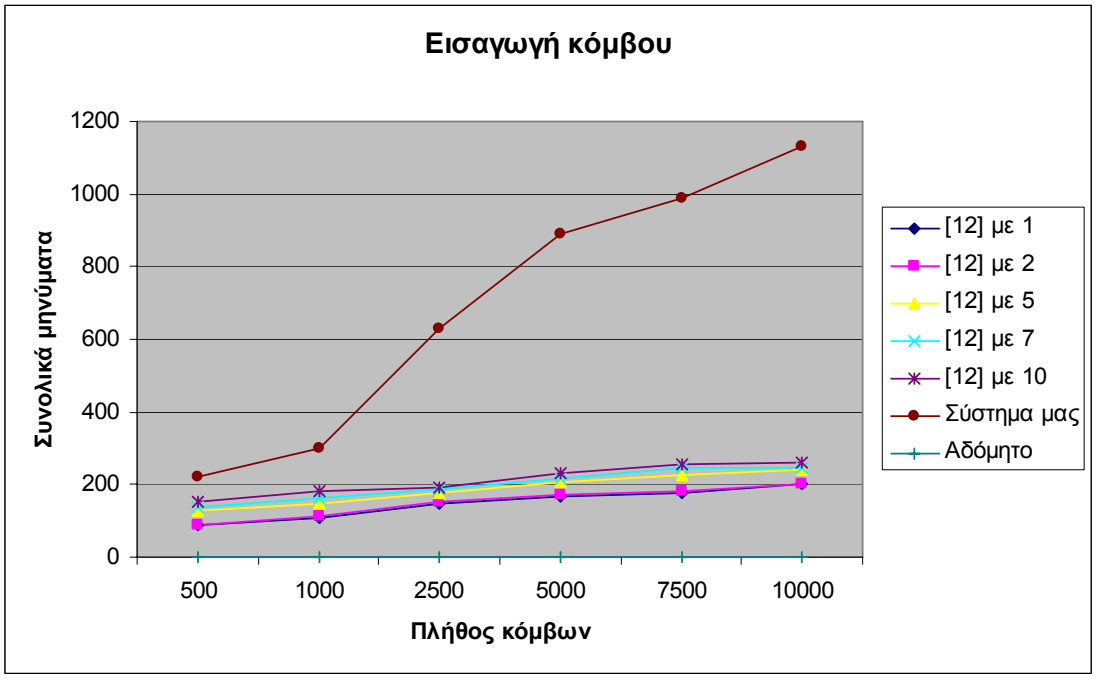
Στα επόμενα πειράματα θα συγκρίνουμε τα 3 συστήματα, στην λειτουργία της εισαγωγής κόμβου. Στο Σχ. 5.21, φαίνεται η σχέση αριθμού κόμβων – αριθμού hops στα 3 συστήματα, για την εισαγωγή κόμβου. Παρατηρούμε, ότι οι χρόνοι εισαγωγής (hops), για διαφορετικά μεγέθη δικτύου, στο αδόμητο σύστημα, είναι ίδιοι και ίσοι με 1 hop, αφού ένας κόμβος για να εισαχθεί στο σύστημα αυτό, απλώς πρέπει να συνδεθεί με έναν κόμβο, που ήδη υπάρχει στο σύστημα. Επίσης, μπορούμε να δούμε τη σχέση του δικού μας συστήματος με αυτό του [12] (για τις 5 διαφορετικές περιπτώσεις που αναφέραμε πιο πάνω). Καταρχήν, παρατηρούμε, ότι όσο περισσότερα είναι τα οριζόντια υποσχήματα που έχει ένα σχήμα, τόσο περισσότερα hops χρειάζονται για να εισαχθεί ένας κόμβος στο σύστημα του [12] (για κάθε μέγεθος δικτύου). Αυτό είναι κάτι που περιμέναμε, αφού, όπως είπαμε, ο χρόνος εισαγωγής ενός κόμβου στο [12], εξαρτάται από τον αριθμό των υποσχημάτων που έχει ένα σχήμα. Επίσης, παρατηρούμε ότι οι καμπύλες, που αφορούν το [12], ακολουθούν την πολυπλοκότητα $K*O(\log^2 n)$. Επίσης, παρατηρούμε ότι το σύστημά

μας, αποδίδει καλύτερα, όταν έχουμε σχήματα που έχουν περισσότερα από 5 οριζόντια υποσχήματα. Όταν έχουμε σχήματα που έχουν λιγότερα από 5 οριζόντια υποσχήματα, τότε οι χρόνοι του συστήματός μας είναι πιο αργοί από το [12]. Ωστόσο, περιμένουμε, ότι τα περισσότερα σχήματα σε ένα πραγματικό δίκτυο, θα έχουν περισσότερα ή ίσα από 5 οριζόντια σχήματα. Αυτό το περιμένουμε, γιατί αν ένα σχήμα έχει 2 ή 3 οριζόντια υποσχήματα, αυτό σημαίνει είτε ότι το σχήμα αποτελείται από 2 ή 3 τριάδες μόνο, ή ότι η μία τριάδα θα υπάγεται στην άλλη. Σε ένα πραγματικό σύστημα, όμως, περιμένουμε ότι θα έχουμε σχήματα με περισσότερες από 3 τριάδες, ενώ θα ήταν και εξαιρετικά απίθανο να έχουμε ένα σχήμα, όπου η κάθε τριάδα του να υπάγεται η μία στην άλλη. Επίσης, παρατηρούμε ότι το σύστημά μας ακολουθεί πολυπλοκότητα $p*O(\log^2 s) + O(h)$, αλλά επειδή για μεγάλο μέγεθος δικτύων ισχύει $h<<s$ κι επειδή το p είναι πολύ μικρό (όπως δείξαμε και σε προηγούμενα πειράματα, που αφορούσαν το σύστημά μας), ουσιαστικά μόνο ο όρος $O(\log^2 s)$ (οι ανανεώσεις των fingertables των υπερσχημάτων) επηρεάζει τους χρόνους εισαγωγής του συστήματός μας.



Σχήμα 5.21 Σύγκριση (σε Hops) των 3 Συστημάτων για την Εισαγωγή Κόμβου

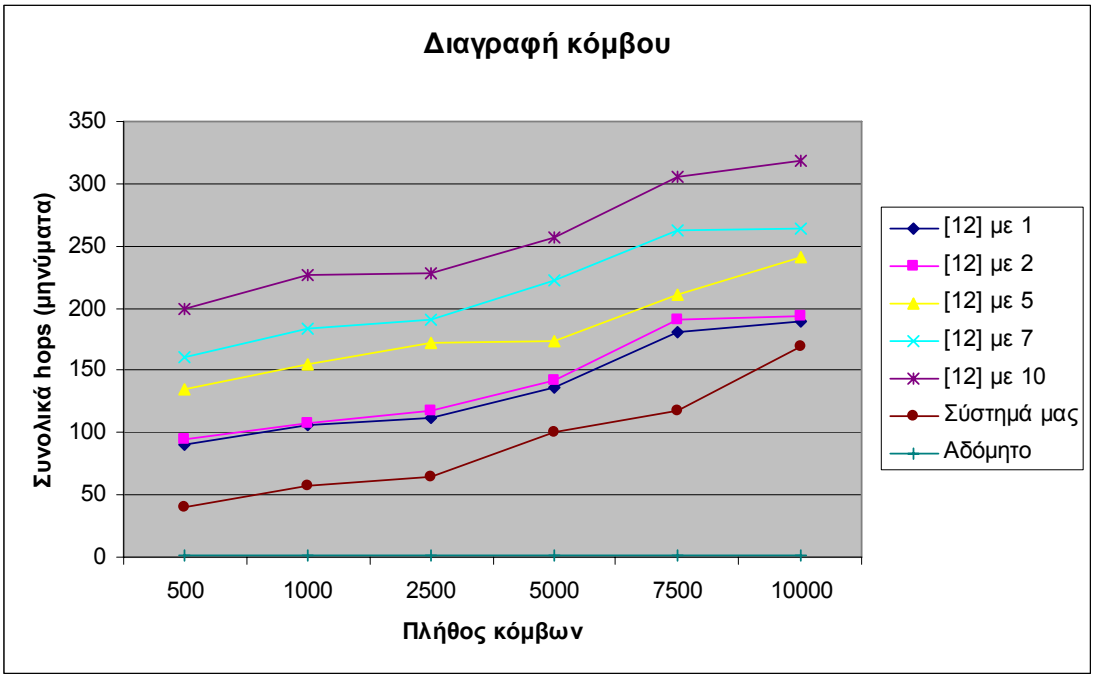
Στο Σχ. 5.22, φαίνεται η σχέση αριθμού κόμβων – αριθμού μηνυμάτων στα 3 συστήματα, για την εισαγωγή κόμβου. Οι παρατηρήσεις που μπορούμε να κάνουμε, είναι ότι κι εδώ το πλήθος μηνυμάτων που απαιτείται στο αδόμητο σύστημα είναι σταθερό και ίσο με 1, αφού μόνο ένα μήνυμα χρειάζεται να ανταλλαχθεί για να συνδεθεί ένας κόμβος με έναν άλλο, σε αυτό το σύστημα. Επίσης παρατηρούμε, ότι το σύστημά μας απαιτεί περισσότερα μηνύματα απ' ότι το σύστημα [12] (και από τις 5 περιπτώσεις), ενώ η καμπύλη του αριθμού των μηνυμάτων ακολουθεί γραμμική συμπεριφορά σε σχέση με τον αριθμό των κόμβων. Αυτό είναι κάτι που περιμέναμε, αφού για να γίνει η εισαγωγή ενός σχήματος στο σύστημά μας, απαιτούνται O(s+h) μηνύματα κι επειδή h<<s (όπως εξηγήσαμε και στην περίπτωση του Σχ. 5.21), ο ουσιαστικός αριθμός μηνυμάτων που απαιτούνται είναι O(s). Αυτή την πολυπλοκότητα ακολουθεί και η καμπύλη του συστήματός μας. Κατά τα άλλα, παρατηρούμε κι εδώ ότι ο αριθμός των μηνυμάτων που απαιτούνται, για την εισαγωγή κόμβου στο [12], αυξάνει, όσο αυξάνει και το πλήθος των υποσχημάτων που έχουν τα σχήματα.



Σχήμα 5.22 Σύγκριση (σε Πλήθος Μηνυμάτων) των 3 Συστημάτων για την Εισαγωγή Κόμβου

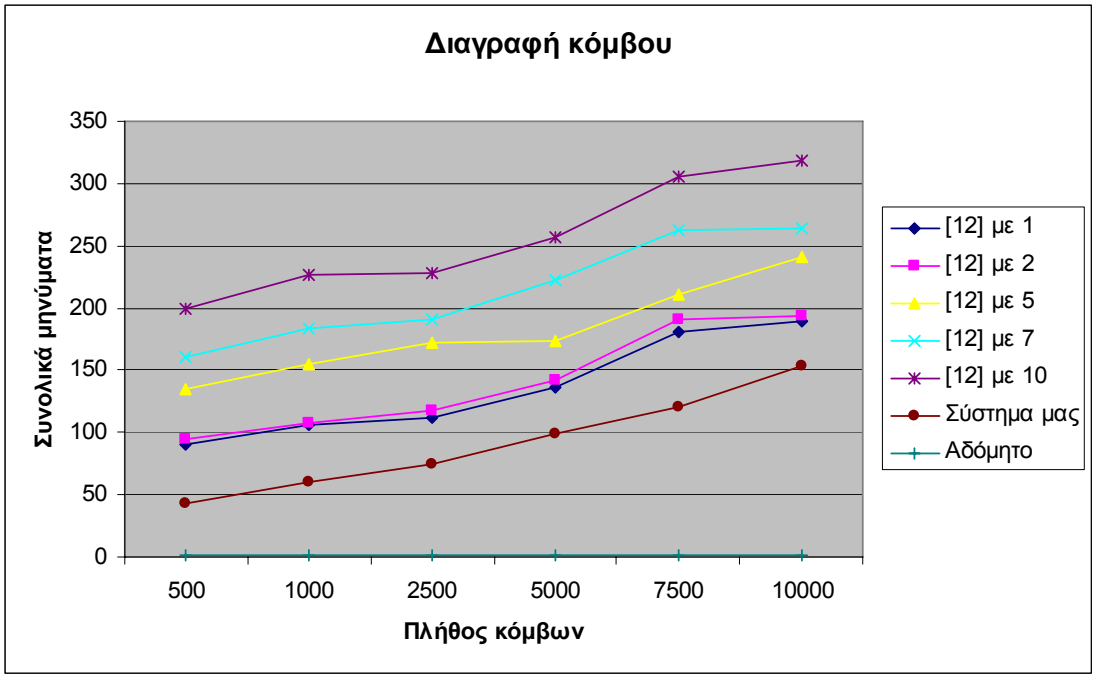
5.4.2. Διαγραφή Κόμβου

Στα επόμενα πειράματα θα συγκρίνουμε τα 3 συστήματα, στην λειτουργία της διαγραφής κόμβου. Στο Σχ. 5.23 φαίνεται η σχέση αριθμού κόμβων – αριθμού hops στα 3 συστήματα, για την διαγραφή κόμβου. Παρατηρούμε, ότι οι χρόνοι διαγραφής του αδόμητου συστήματος είναι μηδενικοί, αφού για να διαγραφεί ένας κόμβος δεν χρειάζεται να κάνει καμία ενέργεια. Επίσης, μπορούμε να δούμε τη σχέση του δικού μας συστήματος με αυτό του [12] (για τις 5 διαφορετικές περιπτώσεις σχημάτων). Παρατηρούμε, ότι όσο αυξάνεται το πλήθος των υποσχημάτων που έχουν τα σχήματα, τόσο αυξάνεται ο χρόνος της διαγραφής κόμβου στο [12], ενώ και οι 5 καμπύλες ακολουθούν την πολυπλοκότητα K*O($\log^2$n). Αντίθετα όμως, με την περίπτωση της εισαγωγής κόμβου, εδώ παρατηρούμε, ότι το σύστημά μας επιδεικνύει καλύτερους χρόνους απ' ότι το [12]. Αυτό συμβαίνει, γιατί από τα πειράματα προκύπτει, ότι ένα υπερσχήμα (πρακτικά) δεν έχει πάνω από 3 παιδιά (επίσης, μην ξεχνάμε ότι h<<s, όπως εξηγήσαμε και στα προηγούμενα πειράματα), άρα ο χρόνος για τη διαγραφή ενός υπερσχήματος δεν είναι μεγαλύτερος από 3*O($\log^2$s). Επίσης,



Σχήμα 5.23 Σύγκριση (σε hops) των 3 Συστημάτων για την Διαγραφή Κόμβου

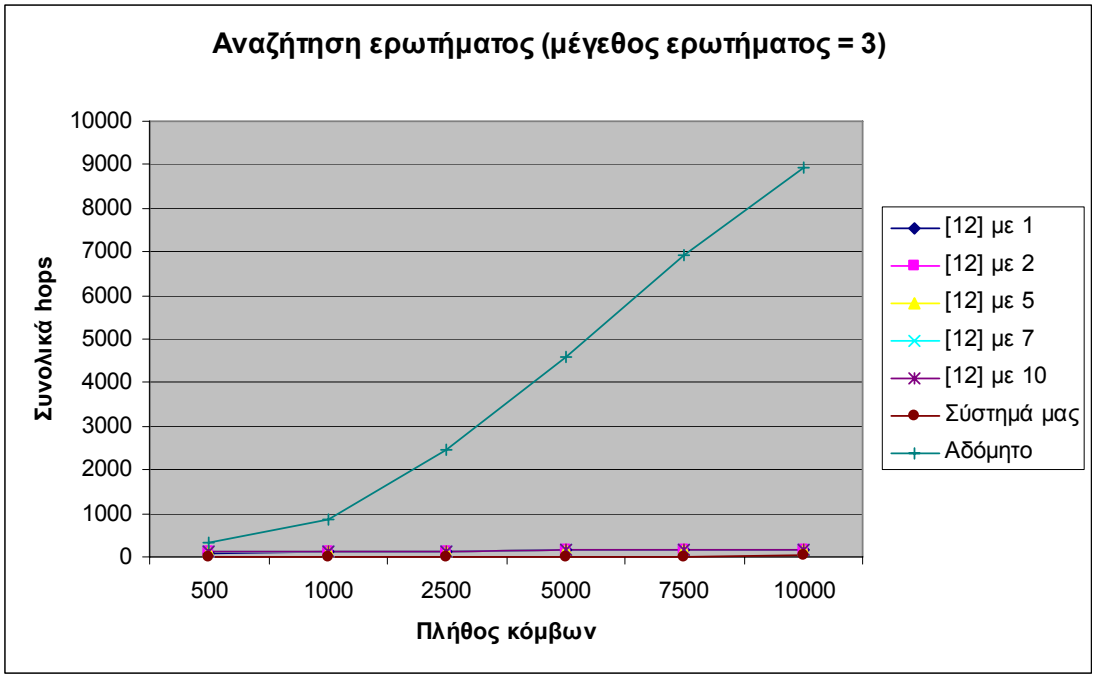
ένας κόμβος που διαγράφεται (στα πειράματά μας οι κόμβοι που διαγράφονταν ήταν τυχαίοι) ενδέχεται να εκδίδει σχήμα που είναι μέρος μιας ιεραρχίας, οπότε σε αυτή την περίπτωση η διαγραφή του είναι τετριμμένη και απαιτεί μόλις 1 hop, κάτι που επηρεάζει, καταλυτικά, τους μέσους χρόνους για τη διαγραφή ενός κόμβου στο σύστημα μας (όπως συμβαίνει και στο πείραμα που παρουσιάσαμε στο Σχ. 10).



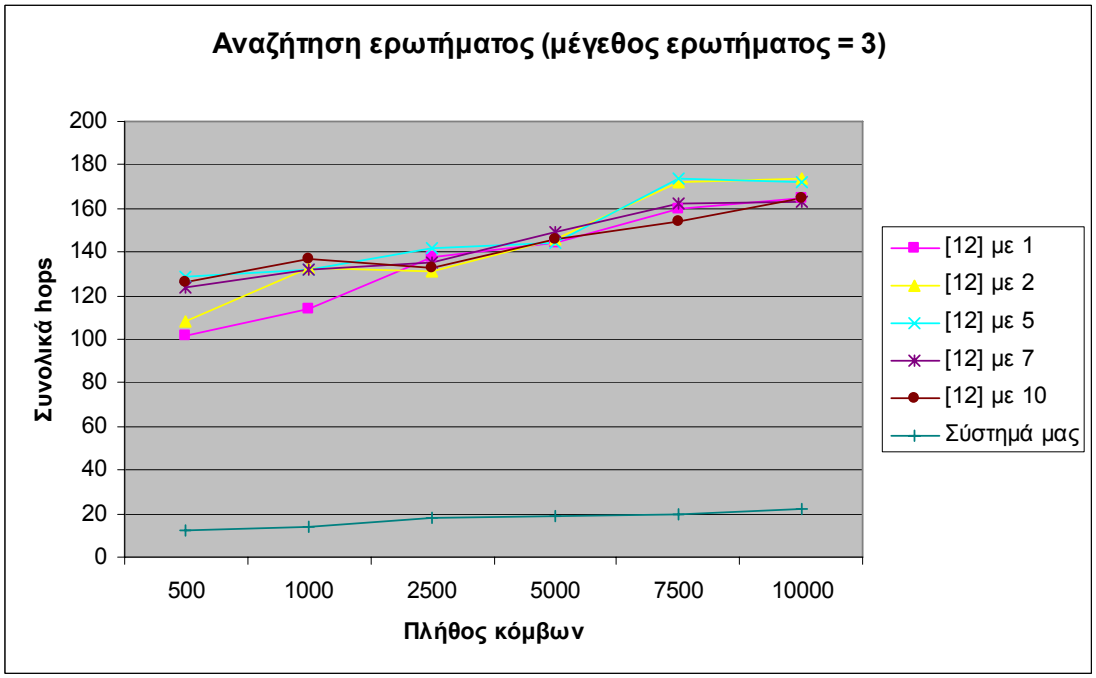
Σχήμα 5.24 Σύγκριση των 3 Συστημάτων για την Διαγραφή Κόμβου

5.4.3. Αναζήτηση Ερωτήματος

Στα επόμενα πειράματα θα συγκρίνουμε τα 3 συστήματα, στην λειτουργία της αναζήτησης ερωτήματος. Στο Σχ. 5.25, φαίνεται η σχέση αριθμού κόμβων – αριθμού hops στα 3 συστήματα, για την αναζήτηση ερωτήματος, όταν το μέγεθος του ερωτήματος είναι 3 τριάδες. Παρατηρούμε ότι το αδόμητο σύστημα ακολουθεί γραμμική συμπεριφορά, αφού πρέπει να στείλει το ερώτημα σε όλους τους κόμβους, με αποτέλεσμα να κάνει broadcast με *πλημμύρα* (κάτι που απαιτεί Θ(n) hops). Στο Σχ. 5.26 μπορούμε να δούμε πιο καθαρά τη σχέση του δικού μας συστήματος και του [12] (και για τις 5 περιπτώσεις του [12]), στην περίπτωση της αναζήτησης ερωτήματος (στην ουσία είναι το ίδιο γράφημα με αυτό του Σχ. 5.25, αλλά σε άλλη κλίμακα, αφού λείπει η καμπύλη του αδόμητου συστήματος). Παρατηρούμε ότι οι

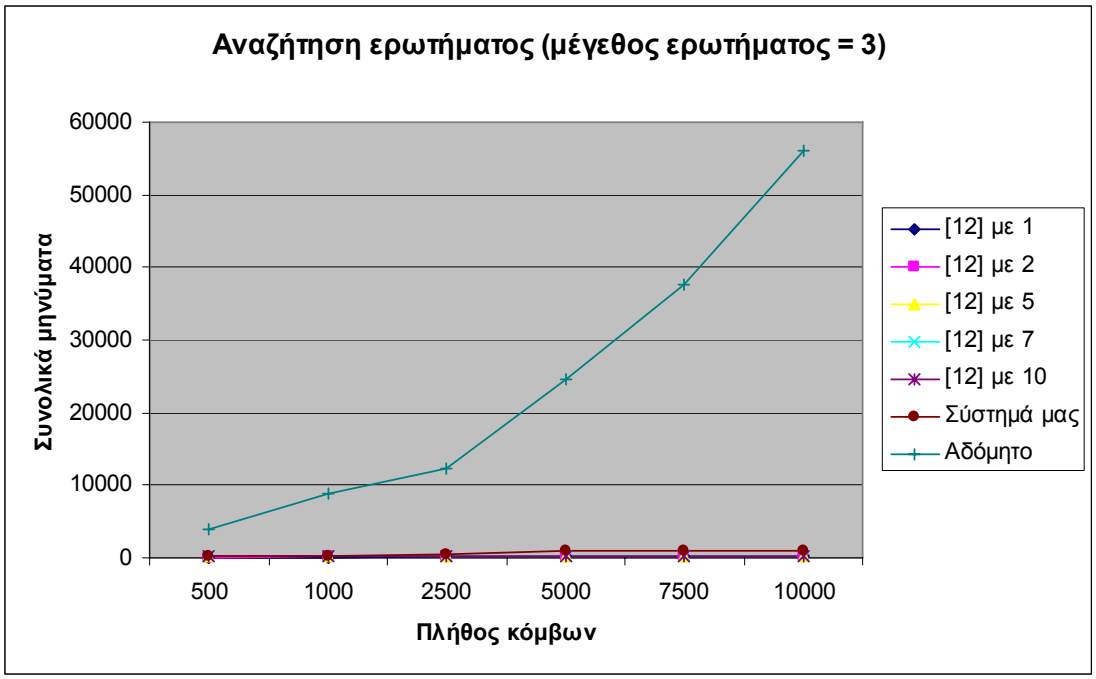


Σχήμα 5.25 Σύγκριση (σε Hops) των 3 Συστημάτων για την Αναζήτηση Ερωτήματος



Σχήμα 5.26 Σύγκριση (σε Hops) του Συστήματός μας με το [12] για την Αναζήτηση Ερωτήματος

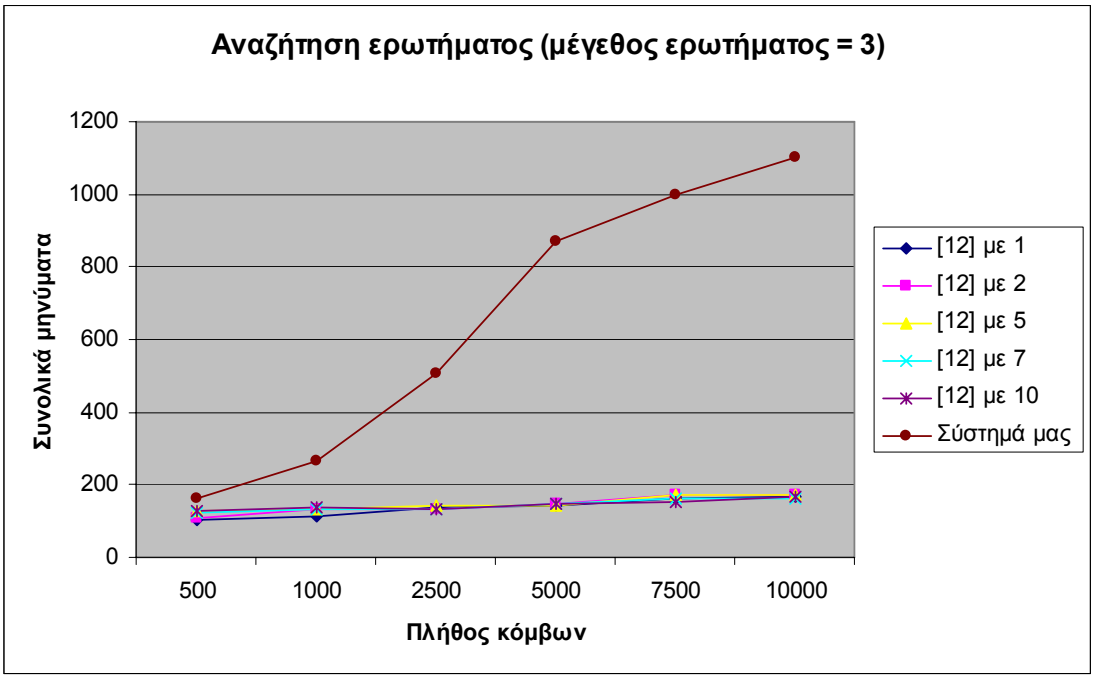
χρόνοι αναζήτησης και για τις 5 περιπτώσεις του συστήματος [12], είναι περίπου ίδιοι και ακολουθούν την πολυπλοκότητα S*O(logn). Αυτό το περιμένουμε, καθώς, όπως έχουμε πει, η αναζήτηση ερωτήματος στο [12], δεν εξαρτάται από το πλήθος των οριζόντιων υποσχημάτων των ερωτημάτων, αλλά από το πλήθος των υποσχημάτων του καθολικού σχήματος, που είναι κάθετα υποσχήματα των σχημάτων των ερωτημάτων (το οποίο επηρεάζεται από το μέγεθος του ερωτήματος, όπως θα δούμε και σε επόμενα πειράματα…). Το σημαντικό εδώ, είναι ότι το σύστημά μας παρουσιάζει πολύ καλύτερους χρόνους από το [12], ενώ ακολουθεί την πολυπλοκότητα O(logs + h).



Σχήμα 5.27 Σύγκριση (σε Πλήθος Μηνυμάτων) των 3 Συστημάτων για την Αναζήτηση Ερωτήματος

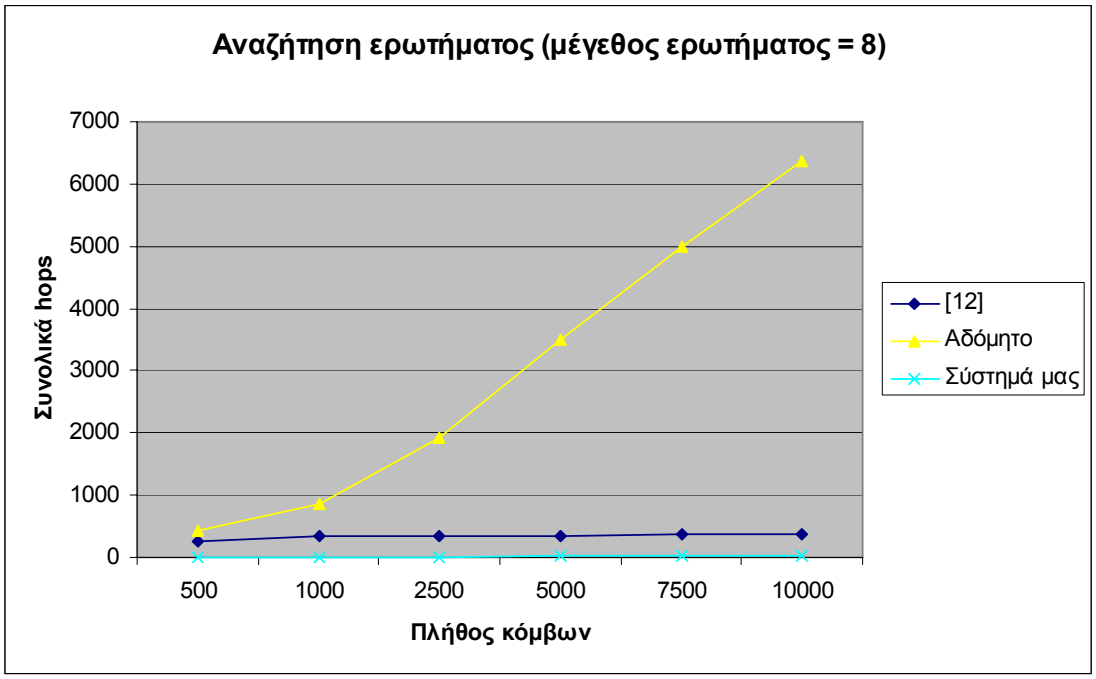
Στο Σχ. 5.27, φαίνεται η σχέση αριθμού κόμβων – πλήθους μηνυμάτων στα 3 συστήματα, για την αναζήτηση ερωτήματος, όταν το μέγεθος του ερωτήματος είναι 3 τριάδες. Παρατηρούμε, ότι το πλήθος μηνυμάτων που απαιτεί το αδόμητο σύστημα είναι πολύ μεγαλύτερο από τα άλλα δύο συστήματα και ακολουθεί εκθετική συμπεριφορά, σε σχέση με το μέγεθος του δικτύου. Αυτό, είναι κάτι που περιμέναμε, αφού, το ερώτημα θα πρέπει να περάσει από όλες τις ιεραρχίες κι αυτό θα γίνει με τη μέθοδο της πλημμύρας, η οποία παράγει πάρα πολλά μηνύματα (Ω(n)). Στο Σχ. 5.28,

μπορούμε να δούμε πιο καθαρά τη σχέση του δικού μας συστήματος και του [12] (και για τις 5 περιπτώσεις του [12]), στην περίπτωση της αναζήτησης ερωτήματος (στην ουσία είναι το ίδιο γράφημα με αυτό του Σχ. 5.27, αλλά σε άλλη κλίμακα, αφού λείπει η καμπύλη του αδόμητου συστήματος). Παρατηρούμε κι εδώ, ότι ο αριθμός μηνυμάτων είναι περίπου ίδιος και για τις 5 περιπτώσεις του [12] κι ότι ακολουθούν πολυπλοκότητα S*O(logn) (οι λόγοι είναι ίδιοι με αυτούς που περιγράψαμε για το Σχ. 5.26). Επίσης παρατηρούμε, ότι το πλήθος των μηνυμάτων που απαιτεί το σύστημα μας είναι πολύ μεγαλύτερο από αυτό που απαιτεί το [12] κι ότι ακολουθεί την πολυπλοκότητα O(s*v). Αυτό το περιμέναμε, καθώς το broadcast που χρησιμοποιούμε στον δακτύλιο παράγει s μηνύματα (όσο και το πλήθος των υπερσχημάτων). Επίσης, στο συγκεκριμένο πείραμα, επειδή το ερώτημα είναι μικρό (έχει μέγεθος 3 τριάδες), τα σχήματα που μπορούν να το απαντήσουν είναι σχετικά λίγα, γι' αυτό κι ο σχετικά μικρός αριθμός σε αριθμό μηνυμάτων (θα περιμέναμε, ίσως, μεγαλύτερο πλήθος μηνυμάτων για το σύστημα μας). Σε επόμενα πειράματα, θα δούμε πως επηρεάζεται η αναζήτηση ερωτήματος στα 3 συστήματα, όταν αυξάνουμε το μέγεθος του ερωτήματος.



Σχήμα 5.28 Σύγκριση (σε Πλήθος Μηνυμάτων) του Συστήματός μας με το [12] για την Αναζήτηση Ερωτήματος

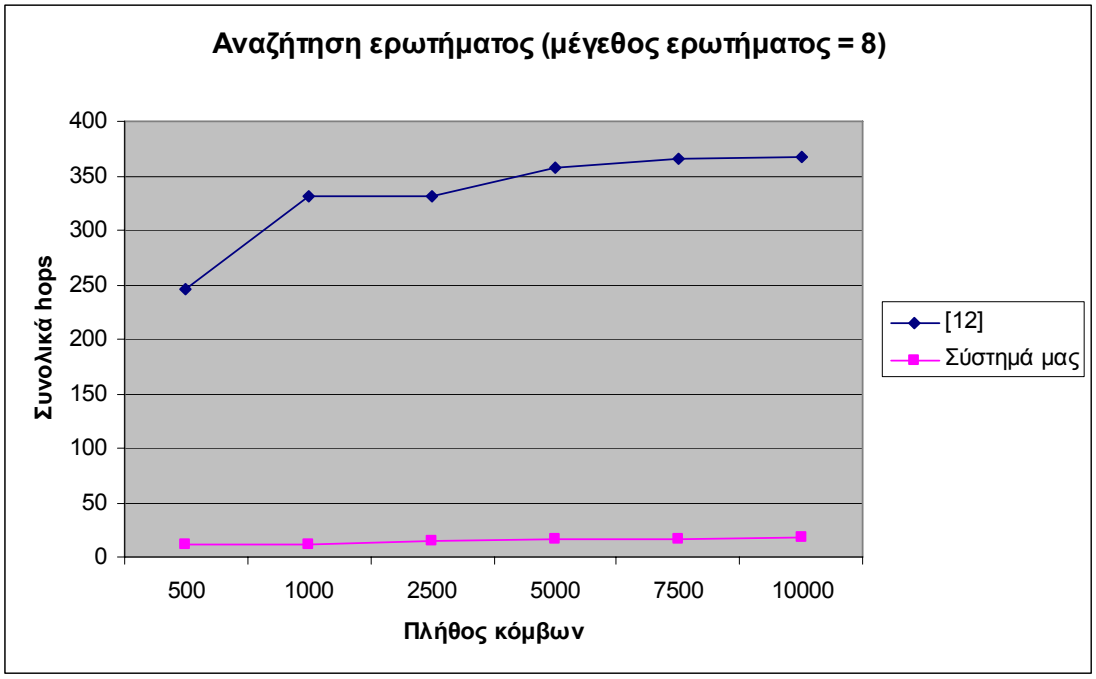
Στα πειράματα που θα δείξουμε στη συνέχεια, φαίνεται η σύγκριση των 3 συστημάτων, στην αναζήτηση ερωτήματος, όταν αυξάνουμε το μέγεθος του ερωτήματος, ενώ το μέγεθος των σχημάτων των κόμβων είναι ίσο με 12 τριάδες (για το σύστημα [12], δεν περιλαμβάνουμε και τις 5 περιπτώσεις, αφού αυτές δεν επηρεάζουν την αναζήτηση σε αυτό, αντίθετα, πήραμε μόνο μια περίπτωση, αυτή για την οποία το πλήθος των οριζόντιων υποσχημάτων των σχημάτων των κόμβων είναι 5 υποσχήματα).



Σχήμα 5.29 Σύγκριση (σε Πλήθος Hops) των 3 Συστημάτων για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 8 τριάδες)

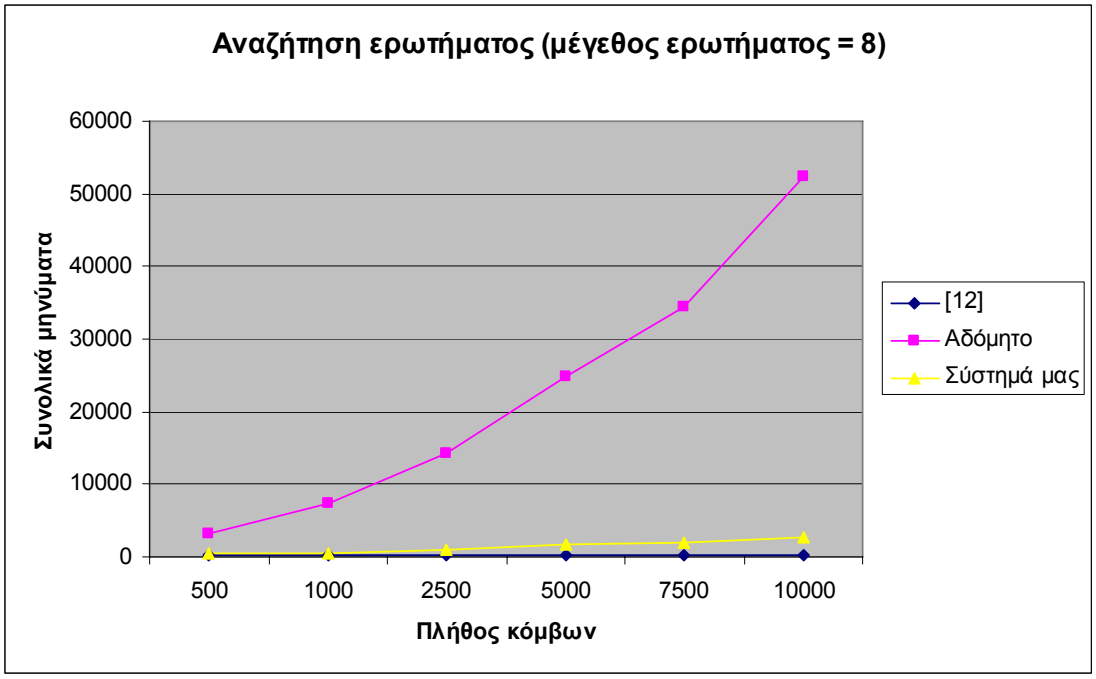
Στο Σχ. 5.29, φαίνεται η σχέση αριθμού κόμβων – αριθμού hops στα 3 συστήματα, για την αναζήτηση ερωτήματος, όταν το μέγεθος του ερωτήματος είναι 8 τριάδες. Παρατηρούμε, ότι το αδόμητο σύστημα, όπως και στην προηγούμενη περίπτωση ερωτήματος, ακολουθεί γραμμική συμπεριφορά, σε σχέση με το μέγεθος του δικτύου, κάτι που κι εδώ περιμέναμε. Στο Σχ. 5.30, μπορούμε να δούμε πιο καθαρά τη σχέση του δικού μας συστήματος και του [12] (στην ουσία είναι το ίδιο γράφημα με αυτό του Σχ. 5.29, αλλά σε άλλη κλίμακα, αφού λείπει η καμπύλη του αδόμητου συστήματος). Παρατηρούμε, ότι οι χρόνοι αναζήτησης του συστήματος [12] ακολουθούν κι εδώ την πολυπλοκότητα S*O(logn). Όπως και για την προηγούμενη

περίπτωση ερωτημάτων, παρατηρούμε κι εδώ, ότι το σύστημά μας παρουσιάζει πολύ καλύτερους χρόνους από το [12], ενώ ακολουθεί κι εδώ, την πολυπλοκότητα O(logs + h). Παρατηρούμε κι εδώ, ότι το μέγεθος του ερωτήματος δεν επηρεάζει δραστικά (έως καθόλου) το σύστημά μας, αφού παρατηρούμε περίπου ίδιους χρόνους με την προηγούμενη περίπτωση ερωτήματος.

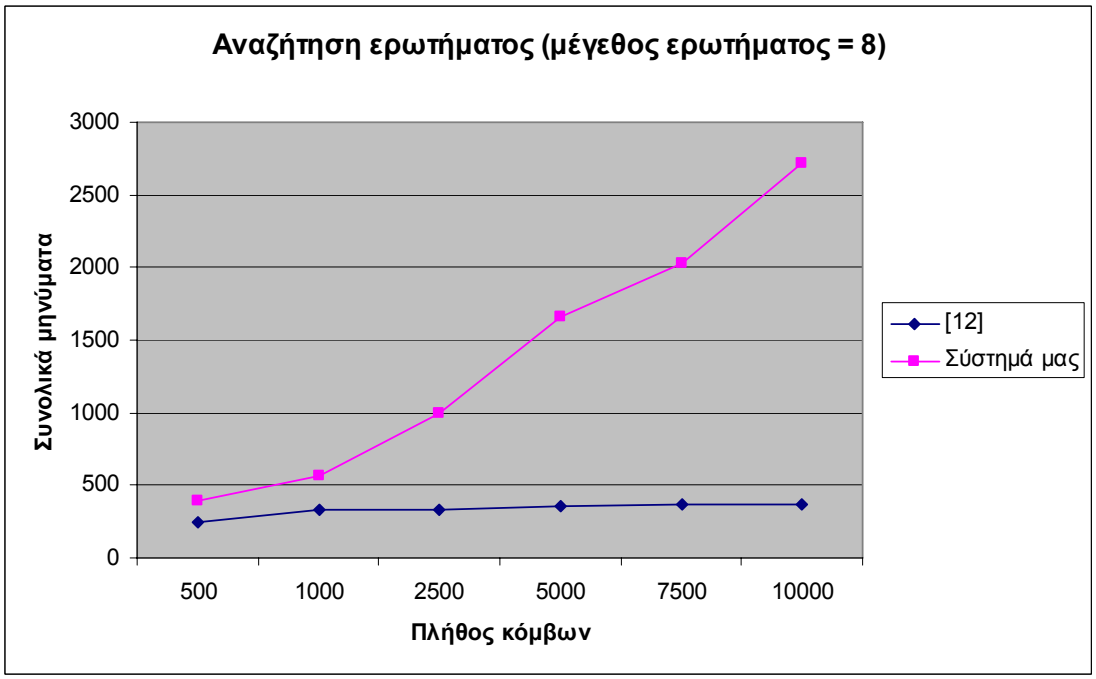


Σχήμα 5.30 Σύγκριση (σε Πλήθος Hops) του Συστήματός μας με το [12], για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 8 τριάδες)

Στο Σχ. 5.31, φαίνεται η σχέση αριθμού κόμβων – πλήθους μηνυμάτων στα 3 συστήματα, για την αναζήτηση ερωτήματος, όταν το μέγεθος του ερωτήματος είναι 8 τριάδες. Παρατηρούμε, ότι το πλήθος μηνυμάτων που απαιτεί το αδόμητο σύστημα, είναι πολύ μεγαλύτερο από τα άλλα δύο συστήματα και ακολουθεί εκθετική συμπεριφορά σε σχέση με το μέγεθος του δικτύου. Αυτό, είναι κάτι που περιμέναμε και σε αυτή την περίπτωση και η εξήγηση είναι ίδια, όπως και στην προηγούμενη περίπτωση ερωτήματος. Στο Σχ. 5.32, μπορούμε να δούμε πιο καθαρά τη σχέση του δικού μας συστήματος και του [12] (στην ουσία είναι το ίδιο γράφημα με αυτό του Σχ. 5.31, αλλά σε άλλη κλίμακα, αφού λείπει η καμπύλη του αδόμητου συστήματος). Παρατηρούμε κι εδώ, ότι ο αριθμός μηνυμάτων για το [12] ακολουθεί πολυπλοκότητα S*O(logn) (οι λόγοι είναι ίδιοι με αυτούς που περιγράψαμε για το

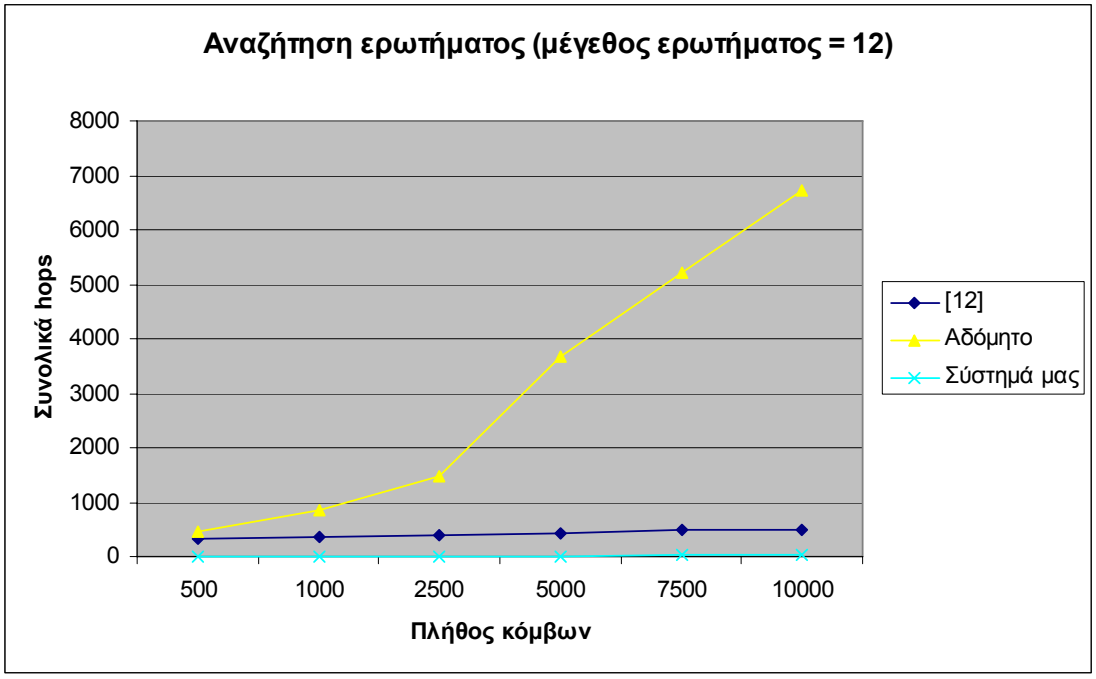


Σχήμα 5.31 Σύγκριση (σε Πλήθος Μηνυμάτων) των 3 Συστημάτων για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 8 τριάδες)



Σχήμα 5.32 Σύγκριση (σε Πλήθος Μηνυμάτων) του Συστήματός μας με το [12], για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 8 τριάδες)

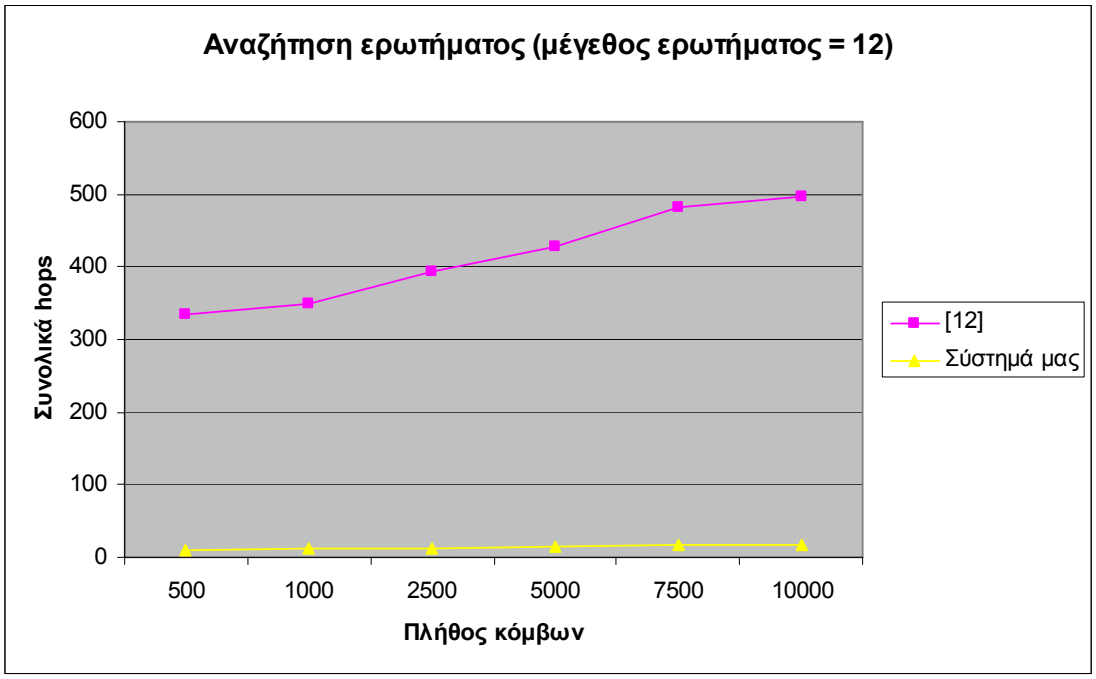
Σχ. 5.26). Επίσης παρατηρούμε, ότι το πλήθος των μηνυμάτων που απαιτεί το σύστημα μας, είναι μεγαλύτερο από αυτό που απαιτεί το [12] κι ότι ακολουθεί την πολυπλοκότητα O(s*ν) (όπως και στην προηγούμενη περίπτωση ερωτήματος).



Σχήμα 5.33 Σύγκριση (σε Πλήθος Hops) των 3 Συστημάτων για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 12 τριάδες)

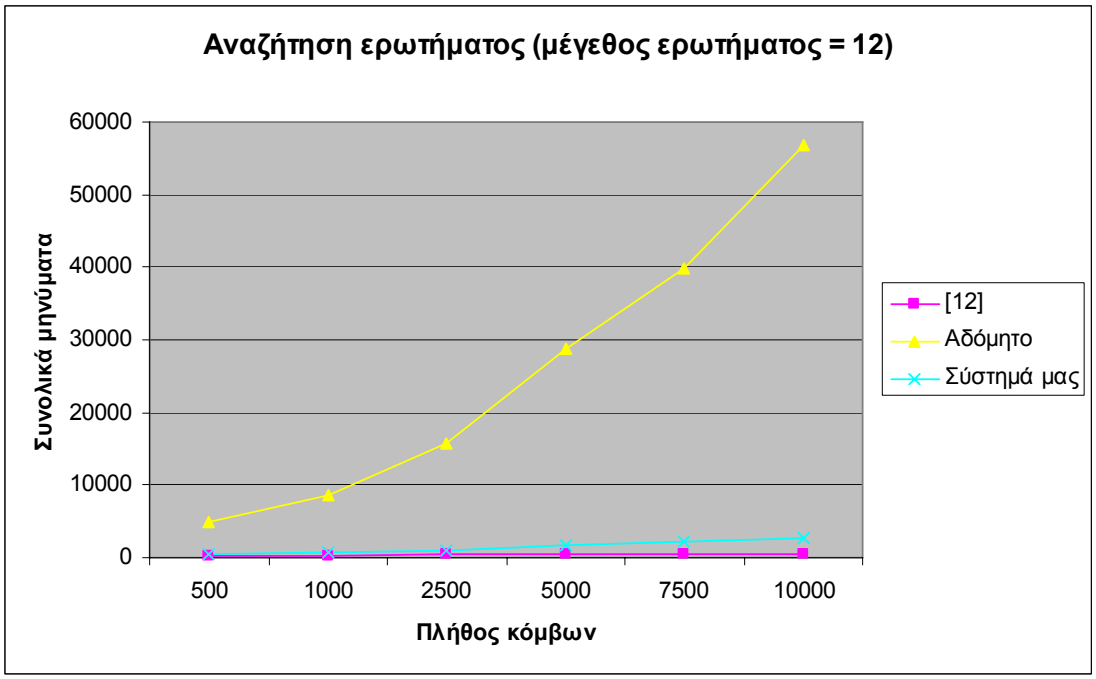
Στο Σχ. 5.33, φαίνεται η σχέση αριθμού κόμβων – αριθμού hops στα 3 συστήματα, για την αναζήτηση ερωτήματος, όταν το μέγεθος του ερωτήματος είναι 12 τριάδες. Παρατηρούμε, ότι το αδόμητο σύστημα, όπως και στις προηγούμενες περιπτώσεις ερωτημάτων, ακολουθεί γραμμική συμπεριφορά, σε σχέση με το μέγεθος του δικτύου, κάτι που κι εδώ περιμέναμε. Στο Σχ. 5.34, μπορούμε να δούμε πιο καθαρά τη σχέση του δικού μας συστήματος και του [12] (στην ουσία είναι το ίδιο γράφημα με αυτό του Σχ. 5.33, αλλά σε άλλη κλίμακα, αφού λείπει η καμπύλη του αδόμητου συστήματος). Παρατηρούμε, ότι οι χρόνοι αναζήτησης του συστήματος [12] ακολουθούν κι εδώ την πολυπλοκότητα S*O(logn). Όπως και για τις προηγούμενες περιπτώσεις ερωτημάτων, παρατηρούμε κι εδώ, ότι το σύστημά μας παρουσιάζει πολύ καλύτερους χρόνους από το [12], ενώ ακολουθεί κι εδώ, την πολυπλοκότητα O(logs + h). Παρατηρούμε κι εδώ, ότι το μέγεθος του ερωτήματος δεν επηρεάζει

σχεδόν καθόλου το σύστημά μας, αφού παρατηρούμε περίπου ίδιους χρόνους με τις προηγούμενες περιπτώσεις.

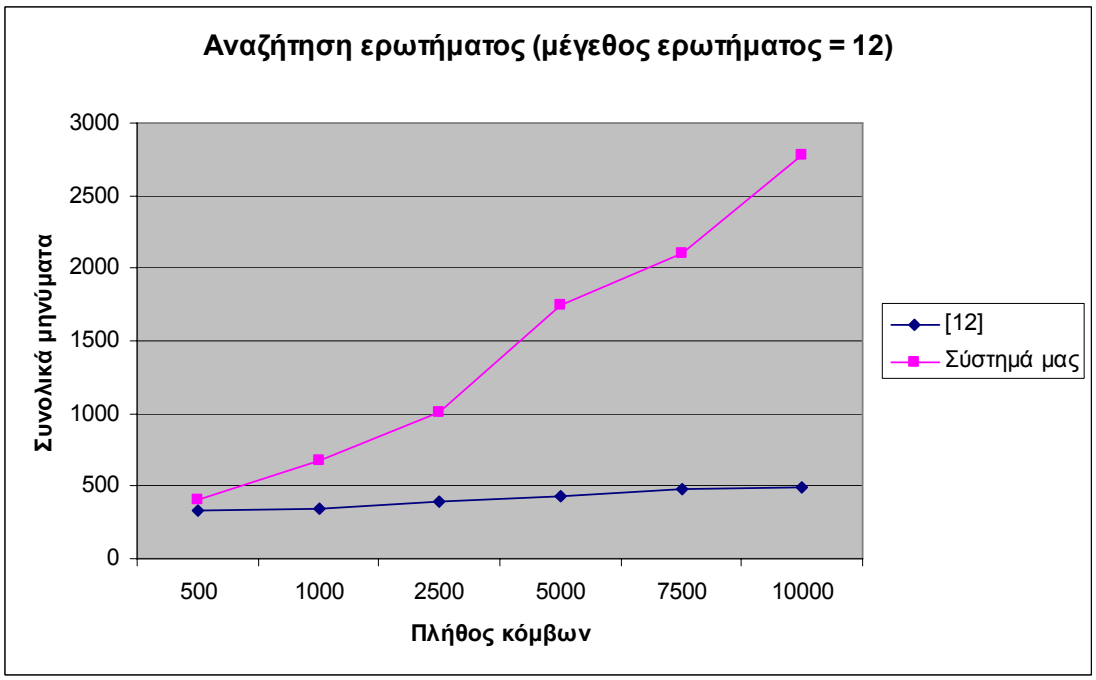


Σχήμα 5.34 Σύγκριση (σε Πλήθος Hops) του Συστήματός μας με το [12], για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 12 τριάδες)

Στο Σχ. 5.35, φαίνεται η σχέση αριθμού κόμβων – πλήθους μηνυμάτων στα 3 συστήματα, για την αναζήτηση ερωτήματος, όταν το μέγεθος του ερωτήματος είναι 12 τριάδες. Παρατηρούμε κι εδώ, ότι το πλήθος μηνυμάτων που απαιτεί το αδόμητο σύστημα, είναι πολύ μεγαλύτερο από τα άλλα δύο συστήματα. Αυτό, είναι κάτι που περιμέναμε και σε αυτή την περίπτωση. Στο Σχ. 5.36, μπορούμε να δούμε πιο καθαρά τη σχέση του δικού μας συστήματος και του [12] (στην ουσία είναι το ίδιο γράφημα με αυτό του Σχ. 5.35, αλλά σε άλλη κλίμακα, αφού λείπει η καμπύλη του αδόμητου συστήματος). Παρατηρούμε κι εδώ, ότι ο αριθμός μηνυμάτων για το [12] ακολουθεί πολυπλοκότητα S*O(logn). Επίσης, παρατηρούμε ότι το πλήθος των μηνυμάτων που απαιτεί το σύστημα μας είναι μεγαλύτερο από αυτό που απαιτεί το [12] κι ότι ακολουθεί την πολυπλοκότητα O(s+v) (όπως εξηγήσαμε και στις προηγούμενες περιπτώσεις ερωτημάτων).



Σχήμα 5.35 Σύγκριση (σε Πλήθος Μηνυμάτων) των 3 Συστημάτων για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 12 τριάδες)



Σχήμα 5.36 Σύγκριση (σε Πλήθος Μηνυμάτων) του Συστήματός μας με το [12], για την Αναζήτηση Ερωτήματος (μέγεθος ερωτήματος = 12 τριάδες)

ΚΕΦΑΛΑΙΟ 6. ΣΧΕΤΙΚΗ ΕΡΓΑΣΙΑ

6.1. Υπερκόμβοι Δομημένοι σε Υπερκύβο

6.1.1. Δομές Δεικτών σε Υπερκύβο

Στο σύστημα [8], οι κόμβοι χωρίζονται σε απλούς κόμβους και υπέρ-κόμβους (*super-peers*). Οι υπέρ-κόμβοι τοποθετούνται σε δομή υπερκύβου και είναι υπεύθυνοι για την αποδοτική αναζήτηση rdf δεδομένων και την διατήρηση της δομής του υπερκύβου. Έτσι, υπέρ-κόμβοι θα πρέπει να είναι τα πιο ισχυρά μηχανήματα από αυτά που έχουμε στο δίκτυό μας. Σε κάθε υπέρ-κόμβο είναι συνδεδεμένοι απλοί κόμβοι, στους οποίους αποθηκεύονται τα rdf δεδομένα. Οι απλοί κόμβοι δεν συνδέονται απευθείας μεταξύ τους, αλλά μόνο με τους υπέρ-κόμβους, σχηματίζοντας τοπολογία αστεριού. Με τη δομή του υπερκύβου εξασφαλίζουμε κάποια από τα πλεονεκτήματά του [6], όπως το γεγονός ότι οι πιο μακρινοί γείτονες θα απέχουν O(logN) βήματα.

Στη συνέχεια, θα δούμε μια δομή που κρατάει ο κάθε υπέρ-κόμβος και βοηθάει τόσο στην εισαγωγή ενός κόμβου και τη σύνδεσή του με έναν υπέρ-κύβο, όσο και στην αποδοτική δρομολόγηση των ερωτημάτων, μόνο σε κόμβους που είναι σχετικοί με τα ερωτήματα (αποτρέποντας τη δρομολόγηση προς όλους τους κόμβους). Τη δομή αυτή, την ονομάζουμε *πίνακα δρομολόγησης υπέρ-κόμβου/κόμβου* (SP/P πίνακας) και κρατάει πληροφορία για τα μετά-δεδομένα σε κάθε κόμβο. Την πληροφορία αυτή, κάθε υπέρ-κόμβος την αποκτά, όταν ένας απλός κόμβος συνδέεται μαζί του και αφού

του στείλει ένα XML μήνυμα σύνδεσης, το οποίο περιέχει την απαιτούμενη πληροφορία. Ένας SP/P πίνακας περιλαμβάνει πληροφορία για:
Δείκτη σχήματος. Μπορεί να υπάρχουν διάφορα σχήματα, άρα κρατάμε ένα αναγνωριστικό σχήματος, καθώς και τους κόμβους που υποστηρίζουν αυτό το σχήμα
Ιδιότητα/ Σύνολα ιδιοτήτων. Ένας κόμβος μπορεί να υποστηρίζει μόνο ένα μέρος ενός σχήματος, έτσι πρέπει να κρατούνται και σύνολα ιδιοτήτων ενός σχήματος, το αναγνωριστικό του σχήματος και οι κόμβοι οι οποίοι υποστηρίζουν αυτές τις ιδιότητες.
Εύρος τιμών ιδιοτήτων. Για ιδιότητες οι οποίες είναι προκαθορισμένα ταξινομημένες, καθορίζουμε το εύρος τιμών που υποστηρίζει κάθε κόμβος.
Δείκτες τιμών ιδιοτήτων. Για ορισμένες ιδιότητες είναι καλό να κρατάμε και δείκτες στις τιμές τους (για παράδειγμα για ιδιότητες που χρησιμοποιούνται συχνά), έτσι ώστε να αποφεύγουμε επιπλέον κίνηση στο δίκτυο.

Παρόμοιοι πίνακες με τους SP/P μπορούν να χρησιμοποιηθούν και για την δρομολόγηση ερωτημάτων μεταξύ των υπέρ-κόμβων. Οι πίνακες αυτοί ονομάζονται *πίνακες δρομολόγησης υπερ-κόμβου/υπερ-κόμβου* (SP/SP πίνακες). Αφορούν δε, μόνο τους υπέρ-κόμβους και είναι στην ουσία αθροίσματα όλων των πινάκων SP/P των απλών κόμβων, με τους οποίους είναι συνδεδεμένος κάθε υπέρ-κόμβος. Ο πίνακας SP/SP ενός υπέρ-κόμβου αποθηκεύεται και στους γειτονικούς υπέρ-κόμβους.

Μια ενημέρωση στην τοπολογία που αναλύουμε, συμβαίνει όποτε ένας κόμβος εισάγεται ή αποχωρεί από το σύστημα ή όταν τα μετά-δεδομένα ενός κόμβου αλλάξουν. Όταν ένας κόμβος φεύγει, τότε όλες οι καταχωρήσεις του SP/P πίνακα (του υπέρ-κόμβου στον οποίο είναι συνδεδεμένος ο αποχωρήσας κόμβος) που αφορούν αυτό τον κόμβο αφαιρούνται. Στην περίπτωση που ένας κόμβος εισάγεται στο σύστημα, τότε τα μετά-δεδομένα του συγκρίνονται με τις καταχωρήσεις του SP/P πίνακα του υπέρ-κόμβου με τον οποίο συνδέεται. Αν τα μετά-δεδομένα υπάρχουν, τότε εισάγετε μόνο το αναγνωριστικό του κόμβου, αλλιώς εισάγονται και όσα μετά-δεδομένα δεν υπάρχουν στον SP/P. Μια ενημέρωση του SP/P πίνακα προκαλεί και την ενημέρωση όλων των SP/SP πινάκων (εναλλακτικά, μπορούμε να μην κάνουμε την ενημέρωση με μία μόνο ενημέρωση ενός SP/P πίνακα, αλλά να περιμένουμε για ένα πλήθος από ενημερώσεις) του υπερκύβου με παρόμοιο τρόπο, με αυτόν που

γίνεται αποδοτικό broadcast σε έναν υπερκύβο [6]. Στην περίπτωση που ένας υπέρ-κόμβος εισαχθεί στο σύστημα, γίνονται οι παρακάτω ενέργειες. Οι υπέρ-κόμβοι τοποθετούνται σε μια κενή θέση του υπερκύβου και κάθε υπέρ-κόμβος είναι υπεύθυνος για μια κενή θέση του υπερκύβου του επόμενου επιπέδου, όπως περιγράφεται στο [6]. Στην περίπτωσή μας, ο υπέρ-κόμβος, που είναι υπεύθυνος για την θέση στην οποία εισάγεται ο καινούριος υπέρ-κόμβος, μεταφέρει σε αυτόν τον μισό SP/SP πίνακα, ενώ οι γείτονές τους ενημερώνονται κατάλληλα. Στην περίπτωση που ένας υπέρ-κόμβος αποχωρήσει από το δίκτυο, μεταφέρει τους πίνακες SP/SP και SP/P στον υπέρ-κόμβο που είναι υπεύθυνος για την κενή θέση που πρόκειται να δημιουργηθεί και ενημερώνονται κατάλληλα οι γείτονές τους.

Η δρομολόγηση των ερωτημάτων στην τοπολογία μας, γίνεται με τον ακόλουθο τρόπο. Ένα ερώτημα μπορεί να δρομολογηθεί αποδοτικά, συγκρίνοντας τα στοιχεία του ερωτήματος με τους πίνακες SP/SP και SP/P, και προωθώντας το, μόνο σε εκείνους τους κόμβους, που μπορούν να υποστηρίξουν τα στοιχεία που εμπεριέχονται στο ερώτημα [8].

6.1.2. Σύστημα Δημοσιοποίησης/Εγγραφής

Η παραπάνω δομή μπορεί να επεκταθεί στο σύστημα που περιγράφεται στο [7]. Το σύστημα αυτό, είναι ένα σύστημα δημοσιοποίησης/εγγραφής των κόμβων στο δίκτυο. Πιο συγκεκριμένα, μας παρέχονται οι εξής δυνατότητες:

Εγγραφής. Οι κόμβοι στέλνουν πληροφορία στο διαδίκτυο, για το είδος των εγγράφουν που θέλουν να ανακτήσουν. Μια εγγραφή είναι της μορφής: $t(S, p_1, O_1) \cap t(S, p_2, O_2) \cap t(S, p_n, O_{1n}) \cap \varphi$, όπου t(S, p, O) είναι μια rdf τριάδα (όπως περιγράφεται στο [3]) με S, p, O το υποκείμενο, το κατηγόρημα και το αντικείμενο αντίστοιχα, ενώ φ είναι περιορισμοί που αφορούν τις τριάδες.

Δημοσιοποίησης. Οι κόμβοι στέλνουν πληροφορία στο διαδίκτυο, για το περιεχόμενο το οποίο θα δημοσιοποιήσουν στους υπόλοιπους κόμβους. Μια δημοσιοποίηση μπορεί να είναι της μορφής: (S, I) (οπότε θα ονομάζεται *δημοσιοποίηση σχήματος*) ή (P, I) (οπότε θα ονομάζεται *δημοσιοποίηση ιδιότητας*) ή $((P_1,V_1), (P_2,V_2), \ldots, (P_n,V_n))$ (οπότε θα ονομάζεται *δημοσιοποίηση ιδιότητας/ τιμής*), όπου S είναι ένα σύνολο

σχημάτων, I είναι το αναγνωριστικό ενός υπέρ-κόμβου, P είναι ένα σύνολο ιδιοτήτων και το ζεύγος (P, V) είναι ένα ζεύγος ιδιότητας/ τιμής.
Ειδοποίησης. Οι κόμβοι ειδοποιούν το δίκτυο, όταν καινούριες πηγές γίνονται διαθέσιμες. Οι πηγές αυτές, προωθούνται προς όλους τους κόμβους, των οποίων το προφίλ εγγραφής ταιριάζει με αυτές.

Η δρομολόγηση των δημοσιοποιήσεων γίνεται με βάση τους πίνακες SP/P και SP/SP, όπως περιγράψαμε στην υποπαράγραφο 6.1.1. Για τη δρομολόγηση των εγγραφών, οι υπέρ-κόμβοι κρατάνε μια ιεραρχία από εγγραφές. Στους θυγατρικούς κόμβους της ιεραρχίας τοποθετούνται εγγραφές, που εμπεριέχονται στους πατρικούς κόμβους, έτσι οι υπέρ-κόμβοι δεν δρομολογούν εγγραφές που εμπεριέχονται σε προηγούμενες εγγραφές (δηλαδή, αν ένας υπέρ-κόμβος λάβει δυο εγγραφές από δυο διαφορετικούς κόμβους και η μια εμπεριέχεται στην άλλη, τότε θα προωθηθεί μόνο η πιο γενική εγγραφή και όχι και οι δύο). Κατά τα άλλα, οι εγγραφές δρομολογούνται όπως οι δημοσιοποιήσεις, με βάση τους πίνακες SP/P και SP/SP. Ένας αλγόριθμος για την δρομολόγηση εγγραφών, φαίνεται στο [7]. Όσον αφορά τη δρομολόγηση των ειδοποιήσεων, γίνονται τα εξής βήματα: όταν μια ειδοποίηση φτάσει σε ένα υπέρ-κόμβο, συγκρίνεται με τη ριζική τοπική εγγραφή, της ιεραρχίας εγγραφών του συγκεκριμένου υπέρ-κόμβου. Αν ταιριάζει, τότε συγκρίνεται με τα παιδιά της ριζικής εγγραφής κ.ο.κ. Για κάθε τέτοια σύγκριση (στην οποία η ειδοποίηση ταιριάζει με τη συγκρινόμενη εγγραφή), η ειδοποίηση στέλνεται σε εκείνο τον υπέρ-κόμβο από την οποία έχει προέλθει η συγκρινόμενη εγγραφή, ακολουθώντας έτσι το μονοπάτι της εγγραφής, κατά την αντίθετη φορά. Ο ολοκληρωμένος αλγόριθμος περιγράφεται στο [7]. Η ανανέωση δε, των δημοσιοποιήσεων γίνεται με τον τρόπο που περιγράψαμε στην υποπαράγραφο 6.1.1 και αφορά τις ανανεώσεις των πινάκων SP/P και SP/SP.

6.1.3. Σύγκριση με το Σύστημά μας

Τα δύο συστήματα που παρουσιάστηκαν στην παράγραφο αυτή, εκμεταλλεύονται το κύριο χαρακτηριστικό του υπερκύβου κι αυτό είναι η διάμετρός του, που είναι logn, αν n είναι οι κόμβοι του υπερκύβου. Κατά συνέπεια, ένα ερώτημα μπορεί να απαντηθεί σε O(logn) hops με O(s) μηνύματα, όπου n είναι ο συνολικός αριθμός από κόμβους (στον υπερκύβο) και s ο συνολικός αριθμός κόμβων. Αυτό, γιατί

χρειαζόμαστε το πολύ logn hops για να φτάσουμε στον πιο μακρινό υπερκόμβο και ένα επιπλέον hop, για να απαντήσουν οι απλοί κόμβοι που είναι συνδεδεμένοι με κάθε υπερκόμβο. Επίσης, χρειαζόμαστε O(logn) hops και O(n) μηνύματα για την εισαγωγή και διαγραφή ενός υπερκόμβου. Στο σύστημα που περιγράφουμε στην παρούσα εργασία, χρειαζόμαστε O(h) + O(logs) hops και O(s*ν) μηνύματα για την απάντηση ενός ερωτήματος, όπου h είναι το βάθος της υψηλότερης ιεραρχίας, s το πλήθος των υπερσχημάτων και ν ο συνολικός αριθμός των κόμβων της μεγαλύτερης ιεραρχίας. Επίσης, στο σύστημά μας απαιτούνται O(h) + p*O(logs) hops και O(s+h) μηνύματα, για την εισαγωγή ενός κόμβου, ενώ για τη διαγραφή ενός κόμβου απαιτούνται c*O(logs) hops και μηνύματα, όπου c ο αριθμός των παιδιών του κόμβου που διαγράφεται. Κατά συνέπεια, τα συστήματα που παρουσιάσαμε στην παρούσα παράγραφο, εμφανίζουν ελαφρώς καλύτερη απόδοση σε ότι αφορά τον αριθμό των hops (εξαρτάται από το πόσο υψηλές είναι οι ιεραρχίες που σχηματίζονται στο σύστημά μας), ενώ εμφανίζουν περίπου ίδια απόδοση, σε ότι αφορά των αριθμό των μηνυμάτων. Το μεγάλο τους, όμως, μειονέκτημα είναι ότι απαιτούν από τους υπερκόμβους να κρατάνε μεγάλο αριθμό πληροφορίας, προκειμένου να διατηρηθεί ο υπερκύβος, αφού θα πρέπει κάθε κόμβος που ανήκει σε ένα επίπεδο του υπερκύβου να κρατάει πληροφορία και για θέσεις του υπερκύβου μεγαλύτερων επιπέδων [5]. Επιπλέον κάθε υπερκόμβος πρέπει να κρατάει δομές με τις κλάσεις και τις ιδιότητες των σχημάτων των κόμβων που είναι συνδεδεμένοι κάτω από αυτόν. Στο σύστημά μας, κάθε υπερκόμβος χρειάζεται να κρατάει μόνο τα finger tables (O(logn) χώρος) και τον successor κόμβο.

6.2. Διαχείριση RDF Δεδομένων σε Αδόμητα Συστήματα

6.2.1. Δομές Δεικτών και Αποδοτικοί Αλγόριθμοι για την Αναζήτηση RDF Δεδομένων

Στο σύστημα [10], απαντάται το πρόβλημα της ολοκληρωμένης πρόσβασης σε rdf βάσεις δεδομένων από μια πρακτική οπτική γωνία. Η δομή που χρησιμοποιούμε, είναι αυτή που περιγράφεται στο σύστημα Sesame, με τη διαφορά ότι εισάγουμε ένα μεσολαβητικό SAIL, μεταξύ των επιμέρους SAIL του Sesame [22] και του SeRQL [23] (η γλώσσα ερωτήσεων του Sesame) parser. Στην αρχιτεκτονική του Sesame, τα ερωτήματα περνάνε από τον parser σε διάφορα SAILs (rdf API), τα οποία είναι

αφαιρετικές δομές της συγκεκριμένης υλοποίησης κάθε βάσης. Κάθε rdf βάση μπορεί να υλοποιηθεί με διαφορετικούς τρόπους κι επειδή θέλουμε να έχουμε πρόσβαση σε διαφορετικές rdf βάσεις, είναι απαραίτητη η χρήση των SAILs. Το πρόβλημα έγκειται, στο ότι σε κατανεμημένες αρχιτεκτονικές, ένα ερώτημα κατανέμεται σε διαφορετικές rdf βάσεις, κάτι το οποίο συνεπάγεται, ότι θα πρέπει να εντοπίσουμε τις σχετικές απαντήσεις, να τις ανακτήσουμε από διαφορετικές πηγές και να τις συνδυάσουμε κατάλληλα. Για το λόγο αυτό, εισάγουμε το *μεσολαβητικό SAIL*. Το μεσολαβητικό SAIL καθορίζει ποιο κομμάτι από το ερώτημα θα πάει σε ποια rdf βάση. Για να απαντήσουμε σε αυτό το ερώτημα, θα χρησιμοποιήσουμε μια δομή δεικτών κι έναν αποδοτικό αλγόριθμο αναζήτησης και ανάκτησης των rdf δεδομένων.

Ο στόχος είναι να προωθούμε ερωτήματα, μόνο σε πηγές πληροφοριών από τις οποίες περιμένουμε ότι θα πάρουμε κάποια απάντηση. Για το λόγο αυτό, χρειαζόμαστε πολύπλοκες δομές δεικτοδότησης. Από τη μια, χρειάζεται να δεικτοδοτούμε πολύπλοκα ερωτήματα, προκειμένου να τα προωθούμε μέχρι την τελική πηγή και από την άλλη, χρειάζεται να μπορούμε να αναγνωρίσουμε υποερωτήματα που χρειαζόμαστε, για να συλλέγουμε ενδιάμεσα αποτελέσματα από κάθε ατομική πηγή. Η δομή που χρησιμοποιείται εδώ, είναι οι *δείκτες συνενώσεων* (*join indices*) [24]. Η ιδέα είναι να δημιουργήσουμε επιπλέον βάσεις δεδομένων, οι οποίες θα κρατούν το αποτέλεσμα μιας συνένωσης, πάνω σε μια συγκεκριμένη ιδιότητα. Σε χρόνο εκτέλεσης, αντί να υπολογιστεί μια συνένωση, το σύστημα προσπελαύνει τη βάση με τους δείκτες συνένωσης, το οποίο απαιτεί λιγότερο χρόνο. Η τελική δομή που παίρνουμε σαν αποτέλεσμα είναι μια ιεραρχία από δείκτες συνένωσης. Το πιο γενικό στοιχείο της ιεραρχίας είναι ένας πίνακας από στοιχεία, που συνδέονται μεταξύ τους με κάποιο μονοπάτι ρ_0, ρ_1, …, ρ_{n-1} μήκους n. Κάθε επόμενο επίπεδο περιέχει όλα τα μονοπάτια ενός συγκεκριμένου μήκους, από 2 μονοπάτια μήκους n-1 στο δεύτερο επίπεδο, έως n μονοπάτια μήκους 1 στο τελευταίο επίπεδο.

Στη συνέχεια, θα δούμε πως οι ιεραρχίες δεικτών συνένωσης μπορούν να χρησιμοποιηθούν, για να αντιμετωπιστεί το πρόβλημα καθορισμού πηγών που περιέχουν αποτελέσματα για ένα υποερώτημα. Γνωρίζουμε, ότι το αποτέλεσμα ενός rdf ερωτήματος (όπως και το ίδιο το ερώτημα) είναι ένα σύνολο υπογράφων (ο

γράφος είναι το rdf μοντέλο), τα οποία αντιστοιχούν σε μία παράσταση μονοπατιού. Πρώτα, αποσυνθέτουμε αυτή την παράσταση σε ένα σύνολο παραστάσεων, που περιγράφουν πιο απλά μονοπάτια. Στη συνέχεια, προωθούμε αυτά τα μονοπάτια στις πηγές που υπάρχει η αντίστοιχη πληροφορία, χρησιμοποιώντας μια ιεραρχία δεικτών συνένωσης, και συνενώνουμε τις απαντήσεις που έχουμε ανακτήσει για να πάρουμε το τελικό αποτέλεσμα. Για να απαντήσουμε σε ένα ερώτημα, πρέπει να βρούμε όλους τους συνδυασμούς υπό-μονοπατιών του ερωτήματος-μονοπατιού που μας έχει δοθεί. Για κάθε έναν από αυτούς τους συνδυασμούς, εντοπίζουμε τις πηγές που περιέχουν τις απαντήσεις, ανακτούμε τα αποτελέσματα και τα συνενώνουμε σε ένα αποτέλεσμα για ολόκληρο το μονοπάτι. Ο αλγόριθμος φαίνεται αναλυτικά στο [10].

Το πλεονέκτημα αυτού του συστήματος είναι ότι απαιτείται μόνο ένα βήμα για τη διαγραφή/εισαγωγή ενός κόμβου. Το μειονέκτημα που παρουσιάζει το συγκεκριμένο σύστημα είναι ο πολύ μεγάλος αριθμός των hops και μηνυμάτων, που χρειάζεται ένα ερώτημα για να απαντηθεί (αφού αν μπορούν να απαντήσουν όλοι οι n κόμβοι, τότε θα πρέπει να αφιερώσουμε O(n) hops και μηνύματα). Επίσης, το σύστημα απαιτεί από τους κόμβους να δεσμεύσουν αρκετό χώρο ($O(s*l^2)$, όπου l το μέγιστο μήκος μονοπατιού στο σχήμα ενός κόμβου και s ο αριθμός των κόμβων, που μπορούν να απαντήσουν στο ερώτημα). Αντίθετα, στο σύστημά μας, χρειαζόμαστε O(h) + O(logn) hops και O(s) μηνύματα για την απάντηση ενός ερωτήματος, όπου h είναι το βάθος της μεγαλύτερης ιεραρχίας, n το πλήθος των υπερσχημάτων (και s είναι ο αριθμός των κόμβων που μπορούν να απαντήσουν σε ένα ερώτημα). Επίσης, στο σύστημά μας απαιτούνται O(h) + p*O(logn) hops και O(n+h) μηνύματα για την εισαγωγή ενός κόμβου (όπου p είναι ο αριθμός των υπερσχημάτων που πρέπει να διαγραφούν), ενώ για τη διαγραφή ενός κόμβου απαιτούνται c*O(logn) hops και μηνύματα, όπου c ο αριθμός των παιδιών του κόμβου που διαγράφεται. Κατά συνέπεια, το σύστημά μας είναι πιο αποδοτικό όσον αφορά την αναζήτηση ενός ερωτήματος και πιο οικονομικό όσον αφορά των απαιτούμενο χώρο, ενώ είναι λιγότερο αποδοτικό όσον αφορά την εισαγωγή/διαγραφή ενός κόμβου.

6.2.2. *Remindin'*

Ο Remindin' [11] είναι ένας αλγόριθμος για την προώθηση και την απάντηση ερωτημάτων, υλοποιημένος πάνω στην πλατφόρμα SWAP [25]. Ο Remindin' χρησιμοποιεί, βασικά, SeRQL [23] ερωτήματα. Ο αλγόριθμος δουλεύει ως εξής. Ας υποθέσουμε ότι ένας χρήστης εκδίδει ένα ερώτημα. Πρώτα, το ερώτημα αξιολογείται τοπικά, πάνω στη βάση του συγκεκριμένου κόμβου. Ταυτόχρονα, το ερώτημα προωθείται προς άλλους κόμβους, για να απαντηθεί από όλο το δίκτυο. Για να το προωθήσουμε, ο Remindin' επιλέγει ένα υποσύνολο κόμβων, οι οποίοι εμφανίζονται πιο πιθανοί από άλλους για να απαντήσουν στο ερώτημα. Υπάρχουν περιπτώσεις, στις οποίες δεν μπορεί να επιλεγεί τέτοιος κόμβος. Σε αυτή την περίπτωση, το ερώτημα διευρύνεται, έτσι ώστε να μπορούν να επιλεγούν όλοι οι κόμβοι. Στη συνέχεια, το αρχικό ερώτημα προωθείται σε ένα υποσύνολο αυτών των κόμβων. Το μήνυμα που περιέχει το ερώτημα, περιέχει ένα μοναδικό αναγνωριστικό και αποθηκεύει τα αναγνωριστικά των κόμβων που επισκέπτεται, προκριμένου να αποφευχθούν οι κύκλοι. Όταν ένας κόμβος δέχεται ένα ερώτημα, θα προσπαθήσει να το απαντήσει και θα αποθηκεύσει ένα δείκτη, που δείχνει στον κόμβο που εξέδωσε το ερώτημα. Η απάντηση επιστρέφεται κατευθείαν σε αυτόν τον κόμβο. Αν ο μέγιστος αριθμός των hops δεν έχει ξεπεραστεί ακόμα, το ερώτημα προωθείται ένα επιλεγμένο σύνολο κόμβων (χρησιμοποιώντας την επιλογή κόμβων που περιγράψαμε). Με την άφιξη των απαντήσεων στον κόμβο που εξέδωσε το ερώτημα, οι σχετικές απαντήσεις επιλέγονται και συμπεριλαμβάνονται στην βάση του κόμβου.

Το πλεονέκτημα αυτού του συστήματος είναι ότι απαιτείται μόνο ένα βήμα για τη διαγραφή/εισαγωγή ενός κόμβου και ο μικρός χώρος που απαιτείται να δεσμεύσουν οι κόμβοι (μόνο τα αναγνωριστικά κάθε ερωτήματος). Το μειονέκτημα που παρουσιάζει το συγκεκριμένο σύστημα (όπως κάθε αδόμητο σύστημα) είναι ο πολύ μεγάλος αριθμός των hops που χρειάζεται ένα ερώτημα για να απαντηθεί (αν μπορούν να απαντήσουν όλοι οι n κόμβοι, τότε θα πρέπει να αφιερώσουμε O(n) hops και μηνύματα). Αντίθετα, στο σύστημά μας, χρειαζόμαστε O(h) + O(logn) hops και O(s) μηνύματα για την απάντηση ενός ερωτήματος, όπου h είναι το βάθος της μεγαλύτερης ιεραρχίας, n το πλήθος των ιεραρχιών και s ο συνολικός αριθμός των κόμβων που μπορούν να απαντήσουν ένα ερώτημα. Επίσης, στο σύστημά μας απαιτούνται O(h) + p*O(logn) hops και O(n+h) μηνύματα για την εισαγωγή ενός

κόμβου (p είναι ο αριθμός των υπερσχημάτων που πρέπει να διαγραφούν), ενώ για τη διαγραφή ενός κόμβου απαιτούνται c*O(logn) hops και μηνύματα, όπου c ο αριθμός των παιδιών του κόμβου που διαγράφεται. Κατά συνέπεια, το σύστημά μας είναι πιο αποδοτικό όσον αφορά την αναζήτηση ενός ερωτήματος, ενώ είναι λιγότερο αποδοτικό όσον αφορά την εισαγωγή/διαγραφή ενός κόμβου και λιγότερο οικονομικό όσον αφορά τον απαιτούμενο χώρο.

6.3. Διαχείριση RDF Δεδομένων και RDF Σχημάτων σε Δομημένα Συστήματα

6.3.1. GridVine

Το GridVine [16] χρησιμοποιεί μια προσέγγιση δύο επιπέδων για να αποθηκεύει και να ανακτά RDF δεδομένα και RDF σχήματα, ξεχωρίζοντας έτσι, το λογικό από το φυσικό επίπεδο. Το λογικό επίπεδο υποστηρίζει λειτουργίες για τη διατήρηση του σημασιολογικού δικτύου (για παράδειγμα, η διαχείριση σχημάτων και η αναζήτηση, με βάση τα χαρακτηριστικά του σχήματος). Το φυσικό επίπεδο είναι ένα δομημένο δίκτυο και πιο συγκεκριμένα το P-Grid. Οι μηχανισμοί που περιγράφονται στη συνέχεια, ακολουθούν τη συντακτική δομή της γλώσσας RDQL [26]. Στο GridVine, τα statements αποθηκεύονται σαν RDF τριάδες και αναφέρονται σε δεδομένα, τα οποία διαμοιράζονται στη δομή του P-Grid. Τα απλά RDF δεδομένα αποθηκεύονται 3 φορές, κατακερματίζοντας το υποκείμενο, το κατηγόρημα και το αντικείμενο ξεχωριστά. Έτσι, η εισαγωγή μιας τριάδας πραγματοποιείται ακολούθως.

Insert(t) ≡ Insert(tsubject, t), Insert(Hash(tpredicate), t), Insert(Hash(tobject), t).

Για να αποθηκεύσουμε RDF Σχήματα, το GridVine καθορίζει δύο νέες μετά-κλάσεις: *P-GridDataItem* και *P-GridDataItemProperty*. Όλες οι κλάσεις κάθε σχήματος, πρέπει να είναι υποκλάσεις της μετά-κλάσης P-GridDataItem και όλες οι ιδιότητες πρέπει να είναι υποϊδιότητες της μετά-κλάσης P-GridDataItemProperty. Με αυτόν τον τρόπο, ο χρήστης μπορεί να εισάγει νέα σχήματα και να τα προσαρμόσει με τα ήδη υπάρχοντα. Έτσι, η εισαγωγή ενός σχήματος πραγματοποιείται με τον ακόλουθο τρόπο.

Insert(rdf schema) ≡ Insert(Hash(p) : class name), rdf schema).

όπου, *hash(p):classname* είναι ένα μοναδικό αναγνωριστικό, το οποίο έχει δημιουργηθεί συνενώνοντας το μονοπάτι *p* ενός κόμβου.

Για να απαντηθεί ένα ερώτημα, όπου δεν είναι γνωστά και τα τρία χαρακτηριστικά της τριάδας, το P-Grid κατακερματίζει τα γνωστά χαρακτηριστικά, για να βρει τον κόμβο, που είναι υπεύθυνος για το συγκεκριμένο χαρακτηριστικό. Στη συνέχεια, ο κόμβος αναζητά στην τοπική του βάση και απαντά με όλα τα ευρήματα. Πρέπει να πούμε εδώ, ότι το GridVine δεν επιτρέπει ερωτήματα, στα οποία δεν είναι γνωστό κανένα χαρακτηριστικό μιας τριάδας.

Επιπλέον, το GridVine χρησιμοποιεί σημασιολογική διάδοση σαν μια μέθοδο προσαρμογής δύο διαφορετικών σχημάτων. Όλοι οι κόμβοι που περιγράφουν τα δεδομένα τους με βάση το ίδιο σχήμα, λέμε ότι ανήκουν στην ίδια σημασιολογική γειτονιά. Κάθε κόμβος έχει τη δυνατότητα να δημιουργεί μια αντιστοίχηση (σύνδεσμο) μεταξύ δύο σχημάτων, δημιουργώντας έτσι ένα δίκτυο που μπορεί να ιδωθεί σαν κατευθυνόμενος γράφος.

Όσον αφορά την εισαγωγή σχήματος, το GridVine πρέπει να αποθηκεύσει ξεχωριστά, κάθε κλάση του συγκεκριμένου σχήματος και κάθε ιδιότητα. Άρα, ο χρόνος εισαγωγής στο GridVine εξαρτάται από το μέγεθος του σχήματος. Έτσι, αν N είναι ο αριθμός των τριάδων ενός Σχήματος, τότε ο χρόνος εισαγωγής (τόσο σε hops, όσο και σε μηνύματα) θα είναι O(3N*logn) hops, αν n είναι ο συνολικός αριθμός των κόμβων του συστήματος (αφού σε N τριάδες υπάρχουν O(2N) κλάσεις και O(N) ιδιότητες). Όπως έχουμε πει, στο σύστημά μας απαιτούνται O(h) + p*O(logs) hops για την εισαγωγή ενός κόμβου στο σύστημά μας, αν h είναι το ύψος της βαθύτερης ιεραρχίας, p ο αριθμός των υπερσχημάτων που πρέπει να διαγραφούν και s είναι ο αριθμός των ιεραρχιών. Επειδή, όπως έδειξαν και τα πειράματα, στο σύστημά μας είναι n>>s (ο συνολικός αριθμός των κόμβων, στο σύστημά μας, είναι πολύ μεγαλύτερος από τον αριθμό των ιεραρχιών). Άρα ισχύει n>s ➔ logn>logs. Εκτός αυτού, το βάθος μιας ιεραρχίας στο σύστημά μας δεν μπορεί να ξεπερνά τον αριθμό των τριάδων του καθολικού σχήματος. Αυτό συμβαίνει γιατί, η μεγαλύτερη ιεραρχία που μπορεί να υπάρξει στο σύστημά μας, είτε θα περιέχει όλα τα σχήματα που υπάγονται οριζόντια στο καθολικό σχήμα (αυτά, δεν θα ξεπερνούν σε ύψος το συνολικό αριθμό των

τριάδων του καθολικού σχήματος) είτε όλα τα σχήματα που υπάγονται κάθετα στο καθολικό σχήμα (τότε, το ύψος της ιεραρχίας δεν μπορεί να ξεπερνά το ύψος του καθολικού σχήματος). Άρα, στη χειρότερη περίπτωση για το σύστημά μας που υπάρχει μια τέτοια ιεραρχία, θα ισχύει h<N κι επειδή ισχύει και logn>logs, έχουμε N*logn> h + logs ➔ O(3N*logn)> O(h) + p*O(logs) (αφού, τα πειράματα, έδειξαν ότι το p δεν είναι μεγαλύτερο από 3). Κατά συνέπεια, το σύστημα μας απαιτεί λιγότερο χρόνο σε hops, από το GridVine για την εισαγωγή ενός σχήματος. Όσον αφορά τον αριθμό των μηνυμάτων, το σύστημά μας απαιτεί O(s+h) μηνύματα. Αν ο αριθμός των ιεραρχιών είναι μικρότερος ή ίσος από logn, δηλαδή αν ισχύει s<logn, τότε το σύστημά μας είναι πιο αποδοτικό και όσον αφορά τον αριθμό των μηνυμάτων (αφού, όπως είπαμε h<N). Στην αντίθετη περίπτωση, δεν μπορούμε να αποκριθούμε με ακρίβεια, για το ποιο σύστημα είναι το πιο αποδοτικό. Όσον αφορά τη διαγραφή ενός κόμβου από το GridVine, ο χρόνος που απαιτείται είναι επίσης O(3N*logn) τόσο σε hops όσο και σε μηνύματα. Στο δικό μας σύστημα, ο χρόνος που απαιτείται είναι c*O(logs) hops μηνύματα, αν c είναι ο αριθμός των παιδιών (του κόμβου που αποχωρεί). Η ανάλυση που κάναμε παραπάνω, ισχύει κι εδώ, με την εξής διαφορά. Για να είναι το σύστημα μας πιο αποδοτικό από το GridVine, θα πρέπει να ισχύει c≤3, σε αντίθετη περίπτωση δεν μπορούμε να αποφανθούμε πιο σύστημα θα είναι, θεωρητικά, το πιο αποδοτικό. Πράγματι, τα πειράματα έδειξαν ότι c≤3, άρα και για τη διαγραφή κόμβου, το σύστημά μας είναι πιο αποδοτικό. Τέλος, όσον αφορά την αναζήτηση ερωτήματος, το GridVine απαιτεί O(k*logn) hops (και μηνύματα), αν k είναι οι κόμβοι, που μπορούν να απαντήσουν στο ερώτημα. Εμείς, χρειαζόμαστε O(logs +h) hops και O(s*l) μηνύματα, αν l είναι ο αριθμός των στοιχείων της μεγαλύτερης ιεραρχίας. Επομένως, στην χειρότερη περίπτωση, όπου όλοι οι κόμβοι μπορούν να απαντήσουν σε ένα ερώτημα, τότε το GridVine θα απαιτούσε O(n*logn) hops (και μηνύματα), ενώ το δικό μας σύστημα θα απαιτούσε O(logs +h) hops και O(n) μηνύματα. Άρα, στη χειρότερη περίπτωση το σύστημα μας λειτουργεί πιο αποδοτικά από το GridVine.

6.3.2. RDFPeers

Στην εργασία [9], οι κόμβοι αυτό-οργανώνονται σε ένα δίκτυο MAAN [27] και κάθε rdf τριάδα (*υποκείμενο*, *προϋπόθεση*, *αντικείμενο*) που εισέρχεται στο δίκτυο,

αποθηκεύεται σε τρία σημεία, χρησιμοποιώντας μια παγκόσμια γνωστή συνάρτηση κατακερματισμού, πάνω στο υποκείμενο, στο κατηγόρημα και στο αντικείμενο. Τα ερωτήματα μπορούν να δρομολογηθούν αποδοτικά (με βάση τους μηχανισμούς του MAAN), εκεί όπου έχουν αποθηκευθεί τα αντίστοιχα rdf έγγραφα. Επίσης, οι υπογραφές για δηλώσεις rdf, αποθηκεύονται στους ίδιους κόμβους, οπότε οι υπογράφοντες μπορούν να ενημερωθούν όποτε μια αντίστοιχη τριάδα εισέρχεται στο δίκτυο. Η εισαγωγή και διαγραφή ενός κόμβου από το σύστημα γίνεται όπως και στο [27], έτσι ώστε να διατηρείται η δομή του συστήματος.

Κάθε rdf τριάδα στο σύστημα αυτό, θα αποθηκευτεί τρεις φορές. Μια φορά με βάση το υποκείμενο, μια με βάση το αντικείμενο και μια με βάση το κατηγόρημα. Κάθε τριάδα αποθηκεύεται στον successor κόμβο του κλειδιού του ζεύγους γνωρίσματος-τιμής, όπως ακριβώς γίνεται και στο MAAN [27], για ένα συγκεκριμένο ζεύγος γνωρίσματος-τιμής. Επίσης, κάθε τριάδα έχει ένα χρόνο λήξης, και ο κόμβος ο οποίος εισάγει μια τριάδα, πρέπει να την ανανεώσει πριν ο χρόνος της λήξει. Αν ο κόμβος ο οποίος είχε αποθηκεύσει την τριάδα, δεν λάβει άλλες ανανεώσεις, τότε διαγράφει την τριάδα. Επιπλέον, οι τριάδες μπορούν να αποθηκεύονται και σε γειτονικούς κόμβους (replication).

Το σύστημα που περιγράφουμε εδώ, μπορεί να απαντήσει σε διάφορους τύπους ερωτημάτων (μέσω ερωτημάτων εύρους πολλαπλών γνωρισμάτων του MAAN). Ο πρώτος τύπος ερωτημάτων περιλαμβάνει τα ατομικά ερωτήματα. Ένα ατομικό ερώτημα είναι μια τριάδα στην οποία το υποκείμενο, το αντικείμενο και το κατηγόρημα μπορεί να είναι μεταβλητή ή ακριβής τιμή, ωστόσο, τουλάχιστον ένα από τα τρία, θα πρέπει να έχει ακριβή τιμή. Έτσι, το ερώτημα θα μπορεί να δρομολογηθεί τουλάχιστο με μια τιμή (με βάση την τριπλή αποθήκευση που περιγράψαμε πιο πάνω). Τα ατομικά ερωτήματα περιγράφονται στο [9].

Ο δεύτερος τύπος ερωτημάτων περιλαμβάνει τα διαζευκτικά ερωτήματα και τα ερωτήματα εύρους. Μπορούμε να επεκτείνουμε τα ατομικά ερωτήματα, χρησιμοποιώντας μια λίστα από περιορισμούς, οι οποίοι περιορίζουν το πεδίο τιμών των μεταβλητών. Το MAAN μπορεί να επιλύσει αποδοτικά ερωτήματα εύρους, χρησιμοποιώντας συναρτήσεις κατακερματισμού που διατηρούν την τοπικότητα

μεταξύ δεδομένων-κόμβων. Επίσης, αντί να ορίσουμε ένα απλό εύρος στο ερώτημά μας, μπορούμε να καθορίσουμε ένα σύνολο από διαζευκτικά εύρη για τιμές γνωρισμάτων. Για παράδειγμα, ένας χρήστης μπορεί να καταθέσει ένα ερώτημα εύρους για την μεταβλητή ?x ε Ένωση$_{i=1...d}$[l_i, u_i]. Ένα τέτοιο ερώτημα θα μπορούσε να επιλυθεί, εκδίδοντας ένα ερώτημα για κάθε διάστημα, και στη συνέχεια υπολογίζοντας την ένωση των αποτελεσμάτων. Μια βελτιωμένη εκδοχή, θα ήταν πρώτα να ταξινομούσαμε τα διαστήματα των ερωτημάτων κατά αύξουσα σειρά. Ένας αλγόριθμος για την επίλυση διαζευκτικών ερωτημάτων εύρους, για την περίπτωση αυτή, περιγράφεται στο [9].

Ο τρίτος τύπος ερωτημάτων περιλαμβάνει συζευκτικά ερωτήματα πολλαπλών κατηγορημάτων. Τέτοια ερωτήματα εκφράζονται σαν διασταύρωση ατομικών ερωτημάτων ή διαζευκτικών ερωτημάτων εύρους, για την ίδια μεταβλητή υποκειμένου. Για την επίλυση τέτοιων ερωτημάτων, χρησιμοποιούμε έναν αναδρομικό αλγόριθμο επίλυσης ερωτημάτων [9], ο οποίος ψάχνει αναδρομικά για υποψήφια υποκείμενα, για κάθε προϋπόθεση και παίρνει την τομή των υποψήφιων υποκειμένων, πριν επιστρέψει το αποτέλεσμα του ερωτήματος.

Το RDFPeers παρουσιάζει αποδοτικούς χρόνους για τη διαχείριση RDF δεδομένων. Ωστόσο, δεν αναφέρεται καθόλου στη διαχείριση RDF Σχημάτων, κάτι που είναι το κύριο στοιχείο της δικής μας εργασίας. Ως εκ τούτου δεν μπορεί να εκμεταλλευτεί τα πλεονεκτήματα ενός δικτύου που στηρίζεται σε κόμβους που προσφέρουν τα δεδομένα τους, δημοσιεύοντας RDF Σχήματα.

6.4. Διαχείριση RDF Δεδομένων και RDF Σχημάτων σε Υβριδικά Συστήματα

Τα *υβριδικά* συστήματα είναι P2P δίκτυα κόμβων, στα οποία κάποιοι κόμβοι δρουν σαν *υπερκόμβοι*. Αυτοί κρατούν περισσότερη πληροφορία από τους απλούς κόμβους και είναι αυτοί, που αναλαμβάνουν την εισαγωγή/διαγραφή τους από το σύστημα, καθώς και την αναζήτηση των ερωτημάτων. Σε αυτά τα συστήματα, οι απλοί κόμβοι είναι συνδεδεμένοι μόνο με τους υπερκόμβους (με έναν από αυτούς) και όχι μεταξύ τους. Οι υπερκόμβοι δε, συνδέονται με άλλους υπερκόμβους τυχαία, σχηματίζοντας

ένα αδόμητο δίκτυο από υπερκόμβους. Στη συνέχεια, θα δούμε δύο υβριδικά συστήματα τα οποία διαχειρίζονται RDF δεδομένα και σχήματα και προτείνουν ένα τρόπο για αποδοτική αναζήτηση ερωτημάτων σε δίκτυα, που δημοσιεύουν τις πηγές μετά-δεδομένων τους με RDF Σχήματα.

6.4.1. PERSINT

Το PERSINT [28] μελετάει το πρόβλημα της ενσωμάτωσης ετερογενών XML και RDF δεδομένων και σχημάτων σε ένα ενιαίο σύστημα. Επίσης, προτείνει μια αρχιτεκτονική για τη διαχείριση αυτών των XML και RDF δεδομένων και σχημάτων σε ένα υβριδικό P2P δίκτυο. Στο σύστημα αυτό, οι υπερκόμβοι περιέχουν καθολικά RDF Σχήματα, ενώ οι απλοί κόμβοι περιέχουν τοπικά δεδομένα και RDF Σχήματα (υποσχήματα των πρώτων). Οι κόμβοι συνδέονται με τους υπερκόμβους μέσω P2P *συσχετίσεων* (*mappings*). Η αρχιτεκτονική έχει 3 κύρια χαρακτηριστικά: (α) Έναν XML-to-RDF wrapper, που χρησιμοποιείται για να μετατρέπει τα XML Σχήματα σε τοπικά RDF Σχήματα, (β) τα τοπικά XML και RDF Σχήματα (που βρίσκονται στους απλούς κόμβους), (γ) τα καθολικά XML και RDF Σχήματα (που βρίσκονται στους υπερκόμβους) και (δ) πίνακες συσχετίσεων, οι οποίοι χρησιμοποιούνται για να αποθηκεύουν τις συσχετίσεις μεταξύ των τοπικών σχημάτων και του καθολικού σχήματος. Όταν ένας κόμβος εισάγεται στο δίκτυο, καταγράφεται και δεικτοδοτείται στον εκάστοτε υπερκόμβο. Για να επιτευχθεί αυτό, ο υπερκόμβος καταρτίζει συσχετίσεις μεταξύ του τοπικού σχήματος του κόμβου και του καθολικού σχήματος του υπερκόμβου. Οι συσχετίσεις αυτές, καταρτίζονται μέσω μιας διαδικασίας σύγκρισης σχημάτων και αποθηκεύονται στον πίνακα συσχετίσεων του κόμβου. Κατά τη διάρκεια της σύγκρισης σχημάτων, το καθολικό σχήμα επεκτείνεται, συνενώνοντας όλα τα τοπικά σχήματα. Η δομή των τοπικών σχημάτων κωδικοποιείται στους συσχετισμούς.

Η αρχιτεκτονική που περιγράψαμε παρέχει δύο μεθόδους αναζήτησης ερωτήματος. Στη μέθοδο *ενοποίησης δεδομένων*, ο χρήστης θέτει ένα ερώτημα (*ερώτημα πηγής*) στο καθολικό σχήμα στον υπερκόμβο. Στη συνέχεια, το ερώτημα αυτό, αναδιαρθρώνεται σε πολλά μικρότερα υποερωτήματα (*ερωτήματα στόχου*), σε κάθε τοπική XML ή RDF πηγή σε κάποιο κόμβο (ένα υποερώτημα για κάθε πηγή). Το

σύστημα εκτελεί κάθε ένα από τα υποερωτήματα ξεχωριστά και συνενώνει τα αποτελέσματα σε μια απάντηση, η οποία στέλνεται στον υπερκόμβο, απ' όπου προήλθε το ερώτημα πηγής. Στη μέθοδο υβριδικού P2P, ο χρήστης μπορεί να θέσει ένα ερώτημα πηγής στην τοπική XML ή RDF πηγή κάποιου κόμβου. Το ερώτημα θα εκτελεστεί στην τοπική πηγή, για να πάρουμε μια τοπική απάντηση. Στο μεταξύ, το ερώτημα αναδιαρθρώνεται σε ένα ερώτημα στόχου σε κάθε άλλο κόμβο στον οποίο καταφθάνει, μέσω *μεταβατικών συσχετίσεων*. Οι μεταβατικές συσχετίσεις είναι συνθέσεις συσχετίσεων από τον αρχικό κόμβο προς τον υπερκόμβο και από τον υπερκόμβο προς τους υπόλοιπους κόμβους-στόχους. Κάθε κόμβος εκτελεί το ερώτημα στόχου που του αναλογεί και επιστρέφει μια απάντηση στον κόμβο απ' όπου ξεκίνησε το ερώτημα πηγής, η οποία καλείται *απομακρυσμένη απάντηση*. Η τοπική και οι απομακρυσμένες απαντήσεις συνενώνονται, για να πάρουμε μια ολοκληρωμένη απάντηση, η οποία στέλνεται στον αρχικό κόμβο. Η μετάφραση των ερωτημάτων γίνεται με βάση τις συσχετίσεις, που υπάρχουν σε κάθε κόμβο, καθώς και από μια συλλογή αλγορίθμων επανεγγραφής των ερωτημάτων.

Το σύστημα που περιγράψαμε, παρέχει έναν τρόπο για τη διαχείριση XML και RDF Σχημάτων σε ένα δίκτυο κόμβων. Ωστόσο, το ίδιο το δίκτυο, πάνω στο οποίο είναι δομημένοι οι κόμβοι, οι οποίοι παρέχουν τα XML και RDF Σχήματα, παρουσιάζει σοβαρά προβλήματα, τα οποία το δικό μας σύστημα τα αποφεύγει. Το μεγαλύτερο μειονέκτημα που παρουσιάζει αυτό το σύστημα είναι ότι λίγοι κόμβοι (υπερκόμβοι) δέχονται πολύ μεγαλύτερο φόρτο απ' ότι οι περισσότεροι κόμβοι του συστήματος. Αυτό θα έχει σαν αποτέλεσμα, αυτοί οι κόμβοι να υπερφορτώνονται με αποτέλεσμα είτε να καταρρέουν είτε να εμφανίζεται το φαινόμενο *bottleneck*, με αποτέλεσμα το όλο σύστημα να υπολειτουργεί. Στο σύστημα μας, το φαινόμενο του υπερβολικού φόρτου που μπορεί να εμφανιστεί σε μια ιεραρχία, μπορεί να λυθεί εφαρμόζοντας την τεχνική εξισορρόπησης φόρτου που περιγράψαμε στην παράγραφο 3.6. Επίσης, ένα ακόμη αρνητικό του PERSINT είναι ότι οι κόμβοι και οι υπερκόμβοι πρέπει με κάποιο τρόπο να γνωρίζουν τις συσχετίσεις, που πρέπει να υπάρχουν μεταξύ τους για να λειτουργήσει το σύστημα (οι συγγραφείς του [28] δεν περιγράφουν πως αυτές μπορούν να αποκτηθούν κατά την ώρα της εισαγωγής ενός κόμβου στο σύστημα). Τέλος, αν υπάρχουν πολλοί υπερκόμβοι, τότε το ερώτημα θα πρέπει να προωθηθεί σε αυτούς, με κάποιο από τους τρόπους που ακολουθούνται στα αδόμητα συστήματα

([2], [13] κλπ.). Αυτό όμως. θα έχει σαν αποτέλεσμα να χρησιμοποιήσουμε ένα μεγάλο μέρος του δικτύου (πολλά μηνύματα, ίσως και παραπάνω από τον σύνολο των κόμβων του συστήματος), για να απαντήσουμε σε ένα ερώτημα, ενώ θα απαιτηθεί και αρκετός χρόνος (πολλά hops, τουλάχιστον όσος και ο αριθμός των υπερκόμβων). Εμείς, στο σύστημά μας έχουμε προνοήσει, ώστε να χρειάζεται λογαριθμικός χρόνος (σε σχέση με τον αριθμό των κόμβων) για την απάντηση ενός ερωτήματος, ενώ και ο αριθμός των μηνυμάτων που απαιτείται είναι μικρότερος από τον συνολικό αριθμό των κόμβων του συστήματος. Αντίθετα, το PERSINT απαιτεί μόνο ένα βήμα για την εισαγωγή/διαγραφή ενός κόμβου από το σύστημα, ενώ στο δικό μας σύστημα απαιτείται τουλάχιστο λογαριθμικό χρόνο (σε σχέση με τον συνολικό αριθμό των κόμβων) για τις δύο αυτές διαδικασίες.

6.4.2. SQPeers

Στο σύστημα που περιγράφεται στην εργασία [29], προτείνεται μια διαδικασία για την αναζήτηση ερωτημάτων, σε ένα δίκτυο κόμβων που παρέχουν τις πηγές μετά-δεδομένων τους με RDF Σχήματα. Κάθε κόμβος που μπορεί να απαντήσει ένα μέρος του ερωτήματος, απαντάει στο υποερώτημα που αντιστοιχεί στο συγκεκριμένο μέρος του συνολικού ερωτήματος. Στη συνέχεια, οι απαντήσεις συνενώνονται για να προκύψει μια τελική απάντηση. Στην εργασία [29], μελετάται η σειρά με την οποία πρέπει να γίνουν αυτές οι συνενώσεις, ώστε η συνολική απάντηση να προκύψει πιο γρήγορα. Επίσης, οι συγγραφείς προτείνουν δύο περιπτώσεις δικτύων, πάνω στα οποία μπορεί να εφαρμοστεί η παραπάνω διαδικασία αναζήτησης ερωτήματος.

Η πρώτη περίπτωση βασίζεται σε ένα σημασιολογικό DHT δίκτυο, όπως το Chord. Σε αυτήν την περίπτωση, κλειδιά από την συνάρτηση κατακερματισμού του DHT δικτύου ανατίθενται σε ολόκληρα υποσχήματα. Για να διατηρηθεί το σημασιολογικό δίκτυο, θα πρέπει να βρεθούν κατάλληλες συναρτήσεις κατακερματισμού (ή κατάλληλοι αλγόριθμοι σύγκρισης σχημάτων), ώστε κόμβοι που έχουν παρόμοια σχήματα να δομούνται σε κοντινές θέσεις στο δίκτυο. Για να απαντηθεί ένα ερώτημα, όλοι οι κόμβοι πρέπει να ερωτηθούν σε διαδοχικά βήματα. Τα αποτελέσματα που θα πάρουμε από κάθε κόμβο, πρέπει να συνενωθούν, ώστε να πάρουμε μια συνολική απάντηση. Το σύστημα DHT που προτείνεται εδώ, μελετάται και περιγράφεται με

ακρίβεια, στην εργασία [12], με την οποία συγκρίνουμε το σύστημά μας και θεωρητικά (στο Κεφάλαιο 4) και πειραματικά (στο Κεφάλαιο 5). Κατά συνέπεια, δεν θα ασχοληθούμε περαιτέρω, με αυτήν τη περίπτωση δικτύου εδώ.

Η δεύτερη περίπτωση βασίζεται σε ένα υβριδικό P2P δίκτυο κόμβων, στο οποίο οι κόμβοι οργανώνονται σε ιεραρχίες, κάτω από συγκεκριμένους υπερκόμβους. Οι θυγατρικοί κόμβοι περιέχου6ν RDF Σχήματα, τα οποία υπάγονται στα RDF Σχήματα των γονέων τους. Κόμβοι οι οποίοι περιέχουν ίδια σχήματα, τοποθετούνται κάτω από τον ίδιο υπερκόμβο, έτσι ώστε κάθε κόμβος να γνωρίζει ποιοι κόμβοι βρίσκονται στην ιεραρχία του. Όταν ένας κόμβος θέτει ένα ερώτημα, που δεν μπορεί να απαντηθεί εξολοκλήρου από τον ίδιο, το ερώτημα προωθείται στον υπερκόμβο που ανήκει ο κόμβος που έκανε το ερώτημα. Αν ο υπερκόμβος δεν περιέχει ή δεν μπορεί να απαντήσει εξολοκλήρου το σχήμα του ερωτήματος, τότε το ερώτημα προωθείται σε έναν από τους υπόλοιπους υπερκόμβους τυχαία, μέχρι να απαντηθεί ολόκληρο το ερώτημα. Αν ένας υπερκόμβος μπορεί να απαντήσει ένα μέρος του ερωτήματος, τότε το ερώτημα προωθείται στην ιεραρχία του υπερκόμβου. Οι απαντήσεις επιστρέφουν στους υπερκόμβους (στους οποίους συνενώνονται). Από εκεί, αποστέλλονται στον υπερκόμβο απ' όπου προήρθε αρχικά το ερώτημα (εκεί συνενώνονται οι απαντήσεις από κάθε υπερκόμβο) και αυτός προωθεί την τελική απάντηση στον κόμβο ο οποίος έκανε το ερώτημα. Όπως παρατηρούμε, η δεύτερη περίπτωση είναι όμοια με την δική μας ιδέα για δημιουργία ιεραρχιών. Ωστόσο, εμείς προχωράμε ένα βήμα πιο πέρα, ενώνοντας τις ιεραρχίες σε μια δομή δακτυλίου, ώστε να μπορούμε να φθάνουμε από μια ιεραρχία σε άλλη σε μικρό χρόνο.

ΚΕΦΑΛΑΙΟ 7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΔΟΥΛΕΙΑ

Στην παρούσα διατριβή ερευνήσαμε την αποδοτική διαχείριση RDF (Resource Description Framework) Σχημάτων σε ένα δίκτυο P2P (peer-to-peer). Προσπαθήσαμε, δηλαδή, να επιτύχουμε την αποδοτική εισαγωγή/διαγραφή κόμβων σε ένα P2P δίκτυο, στο οποίο κάθε κόμβος αναπαριστά τη βάση μετά-δεδομένων του με ένα RDF Σχήμα. Επίσης, ενδιαφερθήκαμε για την αποδοτική αναζήτηση RDF Σχημάτων σε ένα τέτοιο δίκτυο. Πιο αναλυτικά, το σύστημα που περιγράψαμε, βασίζεται σε μια δομή δακτυλίου και σε διαφορετικές ιεραρχίες από RDF Σχήματα. Προσπαθήσαμε να δομήσουμε κόμβους με σχήμα, το οποίο είναι υποσύνολο του ίδιου RDF Σχήματος, στην ίδια ιεραρχία, ενώ το ριζικό σχήμα κάθε ιεραρχίας τοποθετούνταν σε μια θέση στον δακτύλιο. Επίσης, περιγράψαμε κι έναν τρόπο εισαγωγής/διαγραφής ενός σχήματος σε αυτό το σύστημα, καθώς κι έναν τρόπο αναζήτησης ενός ερωτήματος. Τέλος, εφαρμόσαμε μια τεχνική εξισορρόπησης φόρτου για το δίκτυό μας. Στα πειράματά μας, μετρήσαμε την απόδοση του συστήματός μας για διαφορετικές περιπτώσεις δικτύων. Πρώτα για ένα δίκτυο κόμβων το οποίο έχει πολλές ιεραρχίες και λίγους κόμβους σε κάθε ιεραρχία, ύστερα, για ένα δίκτυο το οποίο έχει λίγες ιεραρχίες και πολλούς κόμβους σε κάθε ιεραρχία και για ένα δίκτυο στο οποίο το πλήθος των ιεραρχιών και των κόμβων σε κάθε ιεραρχία είναι τυχαίο. Επίσης, συγκρίναμε το σύστημά μας με ένα σύστημα στο οποίο οι κόμβοι συνδέονται μεταξύ τους τυχαία (αδόμητο σύστημα), καθώς και με ένα σύστημα από την υπάρχουσα βιβλιογραφία. Το σύστημά μας ήταν πιο αποδοτικό όταν το σύνολο των ιεραρχιών ήταν μικρό σε σχέση με τον συνολικό αριθμό των κόμβων, ενώ, όταν στην αντίθετη περίπτωση, παρατηρήσαμε μια μεγάλη αύξηση

στον αριθμό των μηνυμάτων που απαιτήθηκαν για τη εισαγωγή/διαγραφή κόμβου και την αναζήτηση ενός ερωτήματος.

Το σύστημα που περιγράψαμε στην παρούσα διατριβή παρέχει ένα αποδοτικό τρόπο, όσον αφορά το συνολικό αριθμό των hops, τόσο για τη εισαγωγή/διαγραφή ενός κόμβου στο σύστημα, όσο και για την αναζήτηση ερωτήματος. Ωστόσο, παράγεται ένας μεγάλος αριθμός μηνυμάτων (γραμμικός στο πλήθος των κόμβων, για την εισαγωγή ενός κόμβου και για την αναζήτηση ερωτήματος), που οφείλεται στο broadcast, που είναι απαραίτητο να γίνει στο δακτύλιο των υπερσχημάτων. Μελλοντική έρευνα θα μπορούσε να γίνει στη μείωση του πλήθους των μηνυμάτων, που προέρχονται από το broadcast στο δακτύλιο. Δηλαδή, θα μπορούσε να μελετηθεί ένας εναλλακτικός τρόπος, ώστε να τοποθετούνται διαφορετικές ιεραρχίες σχημάτων σε διαφορετικούς ή κοντινούς κόμβους (ίσως, με μια εξειδικευμένη συνάρτηση κατακερματισμού), ώστε η μετάβαση στις ιεραρχίες (στην ουσία, στα υπερσχήματα κάθε ιεραρχίας) να γίνεται σε λογαριθμικό αριθμό μηνυμάτων, σε σχέση με τον αριθμό των ιεραρχιών. Επίσης, η μελέτη και η δημιουργία του συστήματός μας στηρίζεται στην ισχυρή υπόθεση ότι κάθε κόμβος γνωρίζει εξαρχής ένα καθολικό σχήμα και με βάση αυτό εκδίδει το δικό του υποσχήμα. Μελλοντική εργασία θα μπορούσε να γίνει, στη δημιουργία ενός παρόμοιου συστήματος (με ικανοποιητικούς χρόνους εισαγωγής αναζήτησης και διαγραφής), χωρίς την, a priori, γνώση ενός καθολικού σχήματος. Επίσης, πειραματική έρευνα μπορεί να γίνει για να καθοριστεί το χρονικό διάστημα, στο οποίο θα μετράται το πλήθος των ερωτημάτων που δέχτηκε μια ιεραρχία, ώστε να γίνεται πιο σωστά η εξισορρόπηση φόρτου.

ΑΝΑΦΟΡΕΣ

[1] D. Brickley, R. V. Guha, RDF vocabulary description language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema/ (Jan. 2003).

[2] http://www.gnutella.com/

[3] http://www.napster.com/

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. SIGCOMM'01, August 27-31, 2001, San Diego, California, USA.

[5] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A Scalable Content-Addressable Network. ACM SIGCOMM 2001, San Diego, CA.

[6] M. Schlosser, M. Sintek, S. Decker, W. Nejdl, HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy, 2002.

[7] P. Chirita, S. Idreos, M. Koubarakis, W. Nejdl. Publish/Subscribe for RDF-based P2P Networks. 1st European Semantic Web Symposium 2004.

[8] W. Nejdl, M. Wolpers, W.Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst. Super-Peer-Based Routing Strategies for RDF-Based Peer-to-Peer Networks. J. Web Sem., 1(2), 2004.

[9] M. Cai, M. Frank, B. Yan, R MacGregor. A subscribable peer-to-peer RDF repository for distributed metadata management. Web Semantics: Science, Services and Agents on the World Wide Web 2 (2004) 109–130.

[10] H. Stuckenschmidt, R. Vdovjak, G. Houben, J. Broekstra. Index Structures and Algorithms for Querying Distributed RDF Repositories. *WWW2004*, May 17–22, 2004, New York, New York, USA.

[11] C. Tempich, S. Staab, A. Wranik: Remindin': Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors. WWW 2004: 640-649

[12] L. Sidirourgos, G. Kokkinidis, T. Dalamagas. Efficient Query Routing in RDF/S schema-based P2P Systems. 2005, Fourth Hellenic Data Management Symposium (HDMS'05), Athens, Greece, August 25-26.

[13] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. ICS'02, June 2226, 2002, New York, New York, USA.

[14] Worldwide web consortium. Extensible markup language (XML) 1.0 (second edition). http://www.w3.org/TR/REC-xml.

[15] http://www.w3.org/RDF

[16] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, T. Van Pelt, GridVine: Building Internet-Scale Semantic Overlay Networks. International Semantic Web Conference 2004: 107-121.

[17] K. Aberer. P-Grid: A Self-Organizing Access structure for P2P Information Systems. Sixth International Conference on Cooperative Information systems (CoopIS 2001), Lecture Notes in Computer Science, 2172: 179-294, 2001.

[18] S. El-Ansary, L. O. Alima, P. Brand, S. Haridi, Efficient Broadcast in Structured P2P Networks. Work funded by the Swedish funding agency, VIN-NOVA, PPC project, the European IST-FET PERITO project and by the PIRATES project at UCL, Belgium.

[19] http://139.91.183.30:9090/RDF/RBench/index.html.

[20] Magkanaraki A, Tannen V, Christophides V, Plexousakis D (2003) Viewing the Semantic Web Through RVL Lenses. In Proceedings of the 2nd International Semantic Web Conference (ISWC)

[21] Karvounarakis G, Alexaki S, Christophides V, Plexousakis D, Scholl M (2002) RQL: A Declarative Query Language for RDF. In Proceedings of the 11thInternational World Wide Web Conference (WWW), Honolulu, Hawaii, USA

[22] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. The Semantic Web - ISWC 2002, volume 2342 of LNCS, pages 54–68. Springer, 2002.

[23] J. Broekstra. SeRQL: Sesame RDF query language. In M. Ehrig et al., editors, SWAP Deliverable 3.2 Method Design, pages 55.68. 2003.

[24] D. Rotem. Spatial join indices. Proceedings of International Conference on Data Engineering, 1991.

[25] http://swap.semanticweb.org

[26] A. Seaborne. Rdql - a query language for rdf.
http://www.w3.org/Submission/RDQL/

[27] M. Cai, M. Frank, J. Chen, P. Szekely, MAAN: a multi-attribute addressable network for grid information services. Proceedings of the 4th International Workshop on Grid Computing, 2003.

[28] I.F. Cruz, H. Xiao, F. Hsu. Peer-to-Peer Semantic Integration of XML and RDF Data Sources. Third International Workshop on Agents and Peer-to-Peer Computing. (AP2PC 2004), 2004.

[29] G. Kokkinidis, L. Sidirourgos, V. Christophides. Query Processing in RDF/S-based P2P Database Systems. Semantic Web and Peer-to-Peer Springer-Verlag, 2006.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Νικόλαος Κρεμμυδάς γεννήθηκε το 1980, στη Λαμία Φθιώτιδας. Μεγάλωσε στο Καρπενήσι Ευρυτανίας και τελείωσε το Γενικό Λύκειο Καρπενησίου με βαθμό 18,5. Το 1999 ξεκίνησε τις ανώτατες σπουδές του στο τμήμα Πληροφορικής του Πανεπιστημίου Κρήτης, από όπου αποφοίτησε το Φεβρουάριο του 2005, με βαθμό 6,77. Κατά τη διάρκεια των προπτυχιακών σπουδών του εργάστηκε στην εταιρεία κατασκευής δικτυοκεντρικών υπολογιστικών συστημάτων Virtual Trip Ltd (Commercial Manager: Δημήτρης Τσίγκος) από 01/06/2004 – 31/8/2004, ως μέρος της πρακτικής του εργασίας. Επίσης, εργάστηκε στην ομάδα του Mobile Computing Group (υπεύθυνη η Dr Μαρία Παπαδοπούλη) του ΙΤΕ, στο Ηράκλειο Κρήτης, κατά το χειμερινό εξάμηνο του ακαδημαϊκού έτους 2004 - 2005. Από το Φεβρουάριο του 2005 είναι μεταπτυχιακός φοιτητής στο τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων. Από το Φεβρουάριο του 2006 είναι μέλος του Εργαστηρίου Κατανεμημένων Συστημάτων (υπεύθυνη η Dr Ευαγγελία Πιτουρά), του τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.