Supporting Range Queries in Schema-Heterogeneous Peer-to-Peer Systems

CS856 - Web Data Management - Winter 2005

Mohamed Ali Soliman
School of Computer Science
University of Waterloo
m2ali@cs.uwaterloo.ca

Rolando Blanco
School of Computer Science
University of Waterloo
rmblanco@uwaterloo.ca

ABSTRACT

Current data sharing Peer-to-Peer (P2P) systems provide minimal or no support for the processing of range queries in heterogeneous data sources. Query processing is typically performed by selecting peers hosting data that contain the attributes in the query. When the data sources are semantically very close, this attribute-occurrence based selection of participant peers may not be discriminatory enough – since the attributes may be contained in most of the peers. In this work we propose techniques to improve the selection of participant peers when processing range queries. We assume an unstructured P2P system and high semantic commonality among data sources. Our proposal is based on the maintenance of catalogue information at specialised super-peers. Super-Peers store detailed information that describe the data at neighbouring peers, as well as summarised information about other super-peers' catalogues. By using its catalogue, a super-peer is able to identify the neighbouring peers that are relevant to the range conditions specified in a query. The catalogue also allows the super-peer to route the query to other super-peers close to peers that may have data pertinent to the query. We present a storage representation for catalogue information, and specify how the catalogue and summary information are used and maintained. Evaluation of several of the proposed techniques is done by implementing a P2P system using the JXTA framework.

1. INTRODUCTION

P2P systems are massively distributed systems for resource sharing. Made popular by the success of file sharing applications [11, 14, 15], more recent P2P systems attempt to support sharing of structured data [7, 16, 17, 21]. In a P2P system that allows to query structured data, a key issue is the identification of peers hosting data relevant to a given query. The most frequent approach to identify these relevant peers is to determine if the entities and attributes in the query occur at the peers' data sources. In some P2P systems with no data heterogeneity, peer selection is further improved by storing summary information of the values for the attributes in a peer [10]. When the data sources are semantically close and the queries include range conditions, the problem of selecting relevant peers is further complicated. Identifying peers with similar entities and attributes to the ones in the query is no longer an option, since most of the peers will have data sources that relate to the query.

```
SELECT product_dim, prod_cost
  FROM products
 WHERE product_class = 'TV'
   AND product_unit = 'INCHES'
   AND product_dim BETWEEN 17 AND 21
   AND product_cost BETWEEN 100 AND 200
```

Figure 1: Motivating Example

To motivate the discussion of the requirements that a P2P system must meet in order to support efficient execution of range queries, assume a P2P application that allows "store" peers to sell items on the web. Although no global schema is imposed in this example, we assume a high degree of similarity among data sources. A typical query in the system may be the selection of the stores that carry a certain product within a certain price range. Figure 1 shows an example of such a query. For simplicity, we have chosen to use SQL in the example, but the techniques we propose apply to non-relational data sources as well. Furthermore, a test implementation of these techniques uses a XML querying language, and assumes XML-based data sources. When using attribute and entity occurrence, most peers will be identified as relevant to the query in Figure 1. This is due to the close semantic similarity among data sources. One option to improve the peer selection criteria is to maintain range information for the attributes for which range conditions are commonly specified in queries. For example, range information for the attribute *product_cost* could be maintained in the system. In this case, only peers carrying products with prices between \$100 and \$200 would be identified as containing relevant data for the query. If TVs are only a small fraction of the products with prices in the \$100 to \$200 range, the peer identification strategy is still unsatisfactory. Peers selling TVs, as well as peers selling other products in the desired price range, would be identified as relevant. Therefore, to better identify the peers relevant to a query, it is necessary to store not only range information, but information about the attributes on which the range attributes are functionally dependent. In the example, this implies storing the price range for each one of the product classes in the system.

In this work we propose extensions to schema heterogeneous unstructured P2P systems to efficiently support processing of range queries. The core of our proposal is a catalogue framework for the selection of peers relevant to a given range query. When compared to current approaches, our catalogue framework improves query processing performance in systems such that:

- There is high semantic similarity between data sources in the system
- There is a very large number of peers
- Location of the data can not be altered
- A high percentage of queries are range queries
- Range conditions are specified for a small subset of attributes
- Attribute updates happen no more than a few times a week
- Peers providing data have an incentive to participate in the system
- Best effort, good enough answers are acceptable

We assume an unstructured P2P system where peers are connected to super-peers. Super-Peers are specialised peers that store catalogue information, process queries submitted by peers, and route queries to promising peers. Catalogue information at the super-peers is categorised as Super-Peer-to-Peer (SP-to-P) when it describes the data source from a particular peer, or Super-Peer-to-Super-Peer (SP-to-SP) when it summarises the SP-to-P catalogue from another super-peer. SP-to-P catalogue information is used by a super-peer to decide if a neighbouring peer stores data relevant to a query. SP-to-SP catalogue information is used to route a query to a super-peer that may have neighbouring peers with data relevant to a query. The main contributions of our work are:

- A distributed catalogue for the efficient processing of range queries in large P2P systems.
- Techniques for the maintenance and distribution of catalogue information.
- Strategies to balance the load among the super-peers maintaining the catalogue
- Implementation of a test bed P2P system and evaluation of several of the proposed techniques.

The rest of this paper is organised as follows. In Section 2 we present other proposals to support range queries in P2P systems and related work. In Section 3 we present the design of our catalogue, how the catalogue is constructed and maintained, and how it supports the processing of range queries. In Section 4 we describe enhancements to the catalogue that allow the distribution of the load among super-peers. The implementation of a P2P system providing most of the proposed catalogue functionality is presented in Section 5, as well as some experiments evaluating the techniques. Finally, conclusions and future work are presented in Section 6.

2. BACKGROUND AND RELATED WORK

Structured P2P systems excel at exact matching queries, but have hash key distribution and load balancing issues when used to support range queries. Schema heterogeneity is an issue in structured systems as well, since most of these systems assume a single global schema. Semi-structured and unstructured systems seem to be better suited for range queries. But most of the unstructured systems assume low semantic similarity among data sources. Under this assumption, selection of relevant peers based on attribute occurrence is acceptable, even when processing range queries. Integration of heterogeneous data sources, on the other hand, has been one of the main areas of research in data sharing unstructured P2P systems.

2.1 Range Queries in Structured P2P Systems

Most of the work to support range queries in P2P system has been proposed for schema homogeneous structured systems [3, 8, 5, 9, 12, 19]. Gupta et al [12] present a Chord [20] based system where similar ranges are hashed to the same peer using locality sensitive hashing. To avoid increased loads on peers hosting the entries for popular ranges, range data is locally cached by peers. No schema heterogeneity nor multi-attribute ranges are allowed. Sahin et al [19], propose a CAN [18] based system with two dimensions. Ranges for a given attribute define a square bounded by lower/higher values. No schema heterogeneity nor multi-attribute ranges are supported in this work. Another CAN based system with support for range queries is presented by Andrzejak et al [3]. In this proposal nearby ranges map to nearby CAN zones. Interval keepers are responsible for subintervals of the attribute's domain. Each peer reports its current attribute value to the appropriate interval keeper. Single attribute queries are routed to interval keepers by propagating the query in two waves. In the first wave, the query is propagated to neighbours that intersect the query and have a higher interval than the current peer. In the second wave, the query is propagated to the peers that have a lower interval. Although data updates are supported, attribute keepers must be maintained for each single attribute for which a range condition can be specified in a query. Multiple heterogeneous data sources are not allowed in this work.

In the Multi-Attribute Addressable Network (MAAN), Cai et al [9] propose a Chord based system with a locality preserving hashing function that guarantees a uniform hash distribution if data values are uniformly distributed. In MAAN, multiple attributes are supported by having multiple hash functions, one per attribute. Multi-attribute queries are decomposed into multiple sub-queries, each on one attribute. Chord-routing for a sub-query is done using the appropriate attribute hash. No considerations are made in MAAN with regards to support for heterogeneous schemas.

Skip Graphs, a non-DHT based structured system that supports range queries, is proposed by Aspnes et al [5]. Query ranges are naturally supported in skip graphs since values are logically ordered in the Level 0 (the root ring). Skip graphs suffer from scalability issues since there is one peer in the skip graph per attribute value. To overcome this scalability problem, in a more recent work [4] Apsnes et al propose a skip graph based on buckets. Each bucket contains a set of values, and one of the values in the bucket is promoted to the skip graph. A given query is routed to the bucket or buckets that contain the ranges. Buckets are linked together in order, so that it is possible to traverse the buckets once the first value within the range has been identified. Skip graphs based systems exhibit load balancing issues when some few queries are very popular. No work has been done to support heterogeneous schemas nor multi-attribute range queries in skip graph systems.

Yet another structured P2P system that supports range queries is Mercury [8]. In Mercury peers are grouped in virtual hubs, one per attribute. Peers within a hub are arranged into a circular overlay. Each peer is responsible for a continuous range. Peers are linked to all virtual hubs causing scalability problems when there is a large number of attributes. Load balancing is performed by probing peers and inviting lightly loaded peers to join heavily loaded hubs. There is no support for heterogeneous schemas in Mercury.

A catalogue framework for structured P2P systems is proposed by Galanis et al [10]. The catalogue is distributed on a Chord based system and is used to locate data stored in XML repositories. Although there is mention of both structural and attribute value summaries in the catalogue, most of this work deals with the representation of structural summaries only.

2.2 Range Queries in Unstructured P2P Systems

Several unstructured systems provide support for heterogeneous or multiple schemas [16, 17, 21]. Our proposal to store and maintain catalogue information in super-peers is inspired by the Edutella system [16]. Our work differs from Edutella's in that we explicitly support functional dependency and range information. Edutella's support for ranges relies on schema-specific knowledge, where values for an attribute imply a range for other attributes. For example, in a data repository storing document information, the value of an attribute "document category" may imply a range of values for the attribute "document subcategory". In this example, Edutella would consider a condition *document_category = 'DATABASES'* as specifying a specific range of values for the attribute *document_subcategory: DISTRIBUTED, CENTRALISED, RELATIONAL, NETWORK, HIERARCHICAL, ..., etc.* Another difference between our proposal and Edutella's is the support for multiple heterogeneous data schemas. Edutella supports a predetermined number of schemas, while we do not impose a limit in the number of schemas in the system – although we do assume that the schemas are semantically close. The way catalogue information is maintained among super-peers also differs in our proposal. We propagate catalogue information to super-peers based on a need-to-know basis. Catalogue information is cached while the information is of interest to the super-peer (i.e. while there are queries running at the super-peer that require the cached catalogue information from the other super-peer). In Edutella, catalogue information at a super-peer is propagated to all other super-peers.

Our approach to integrate heterogeneous data sources is similar to PeerDB's [17], where attribute name similarity is used to determine schema mappings. Peers register alternate names for the attributes in their schema, and these alternate names are used to identify attribute equivalence even if their names differ – assuming at least one of their alternate names match. In PeerDB human intervention is required to validate schema mappings done by the system. In our proposal, given our assumption of semantic similarity among schemas, we do not require human intervention and consider schema mappings deduced by the system as valid. PeerDB does not provide explicit support for range queries. Hence, any peer with attributes relevant to a query will be visited irrespectively of the value or range conditions specified in the query.

Piazza [21], is another unstructured P2P system that supports heterogeneous data sources. Queries submitted to the system are reformulated based on storage descriptions that refer to relations/attributes in another peer schema. No specific considerations are made in Piazza with regards to range queries. Other proposals focusing on the integration of heterogeneous data sources are the Chatty Web [2] and the Local Relational Model (LRM) [7]. In th Chatty Web, peers route queries to other peers for which no schema mappings exists. Responses are then analysed to decide the semantic similarity among the data sources. Based on this analysis, candidate mappings among the schemas are defined. In the LRM, queries are reformulated based on coordination formulas that describe the data sources on other peers. Neither the Chatty Web nor the LRM attempt to improve processing of range queries. In Piazza, the Chatty Web and LRM, mappings among different data sources are made by peers when there is a motivation to share data ("on a need-to-interact basis"), and are maintained by the peers themselves. Contrasting with our proposal, in these systems queries are executed at the peers themselves, rather than at specialised super-peers. Moreover, queries can not be propagated to other peers unless semantic mappings exist among intermediate peers.

3. CATALOGUE SUPPORT FOR RANGE QUERIES

Given the range query in Figure 1, a catalogue framework should be able to: (1) identify the peers with data repositories containing the attributes in the query (*product_class*, *product_unit*, *product_dim*, and *product_cost*), and from those peers, (2) determine the ones for which the query conditions hold. The first problem relates to the integration of heterogeneous data sources. The second problem relates to the support of range conditions.

In our catalogue framework, we assume an unstructured P2P system with peers connected to super-peers. Super-Peers are specialised, high capacity peers, that store catalogue information and process queries. A peer wishing to execute a query, submits the query to its connected super-peer for processing. The super-peer uses the catalogue to route the query to promising peers – the peers that may store data relevant to the query. As shown in Figure 2, there is no direct connection between peers, and peers communicate to each other via super-peers (peers are labelled P in the Figure). We make no assumptions with regards to the topology of the connections among super-peers, nor impose a protocol that peers must follow to attach to a super-peer when first joining the system. The catalogue at each super-peer stores SP-to-P (super-peer to peer) and SP-to-SP (super-peer to super-peer) information. The SP-to-P information describes the data sources and attribute value ranges of the peers connected to the super-peer. The SP-to-SP information allows a super-peer to identify other super-peers close to data sources that may contribute data to the results of a query. Logically, the SP-to-SP part of the catalogue can

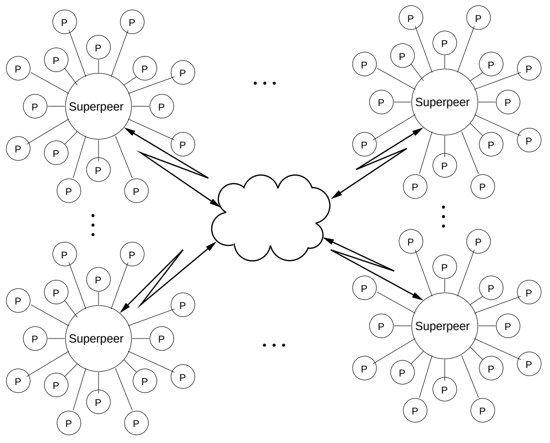

Figure 2: Super-Peer P2P System

be considered an aggregation of the SP-to-P information. Assuming a large amount of data sources in the system, implementing the SP-to-SP part of the catalogue as the union of the SP-to-P information is not a viable option due to scalability issues. In section 3.2, we present different options to address this issue, including the use of summaries, caching, and on-demand utilisation.

We now focus on how the SP-to-P and SP-to-SP parts of the catalogue support the integration of heterogeneous data sources and the processing of range queries.

3.1 SP-to-P Catalogue Information

The SP-to-P catalogue allows a super-peer to determine which peers have data sources compatible with a query; and from these peers, it allows the super-peer to identify the ones with data in the ranges specified by the query.

Upon joining the system, peers register the entities and attributes that they want to make available for querying, as well as the data ranges for the attributes for which range conditions are commonly specified in queries. A peer submits its registration to the super-peer it is connected to. The registration is valid for a time to live TTL interval and peers must re-register their data sources before the TTL expires if they want to keep their data available to the system. The TTL is set by the peers and must fall within a valid range as specified individually by each super-peer. We refer to the description of the entities and attributes in the registration as the *schema description*, and the specification of range information as the *range descriptions*.

3.1.1 Schema Descriptions

When a super-peer receives a query Q for processing, it must first identify the peers attached to the super-peer that contain the entities and attributes mentioned in the query. These entities and attributes in Q can be represented as a set T of pairs (E_q, a_q) where E_q is the name of the entity, and a_q is the name of the attribute in E_q. For the query in Figure 1, the set T is { *(products, product_class), (products, product_unit), (products, product_dim), (products, product_cost)* }.

The names used in Q to refer to entities and attributes may not match the names used by the peers. Hence, peers are advised to register not only the names of the entities and attributes in their data sources, but also synonyms or alternate names that may be used in queries to refer to the same data. In the program implementing the catalogue (Section 5), we refer to these synonyms/alternates for an attribute as its *keywords*. For each neighbouring peer P and each entity E_p in the peer's data source, a super-peer keeps the names registered by P for E_p and each of its attributes a_p. We use the function $syns(E_p)$ to refer to the names of the entity E_p for peer P, and $syns(E_p, a_p)$ to refer to the names of the attribute a_p of entity E_p in peer P. For simplicity, we assume that the name of the entity is in the synonyms set for the entity ($E_p \in syns(E_p)$), and that the name of attribute is in the synonyms set for the attribute ($a_p \in syns(E_p, a_p)$). For the example in Figure 1, possible sets of synonyms for the entity *products* in a peer are: *syns(products)* = { *products, sale_items, article, merchandise* }. The synonyms for the attribute *product_class* in the same peer are *syns(products, product_class)* = { *product_class, product, prod_type, prod_class, prod_group, item* }

In a real-world deployment of a framework as the one we propose, we expect super-peers to provide feedback to peers with regards to the names commonly used in the queries. This feedback would allow peers to improve the alternate names for their attributes and entities. Another possible deployment scenario is one in which queries are predetermined by the applications running in the system. Hence the assumption in this scenario is that there are few or no ad-hoc queries. The names used by the applications to refer to entities and attributes could be provided to peers. Peers would then specify the name equivalences between the application queries and their own data sources.

Given a query Q, the super-peer must be able to extract the set T of entity/attribute pairs, and based on the schema information registered by the peers, identify the set Ps of peers hosting the entities and attributes in T. Formally, assuming the existence of a function $sim\_e(s_1, s_2)$ able to decide if the strings s_1 and s_2 are similar, the set Ps contains the peers P for which:

$$\forall (E_q, a_q) \in T : \exists E_p :$$
$$\Big((\exists S_p \in syns(E_p) : sim(E_q, S_p)) \wedge \quad (3.1)$$
$$(\exists S'_p \in syns(E_p, a_q) : sim(a_q, S'_p) \Big) \quad (3.2)$$

Condition (3.1) specifies that the peer has an entity E_p with a synonym similar to the entity E_q in the query. Recall that the name of the entity E_p is itself in the synonym set, so the names of E_p and E_q may be the same. Condition (3.2) requires that each query attribute a_q of E_q must be similar to a synonym of an attribute in E_p.

The implementation of the similarity function sim may be as simple as a character-by-character comparison, or it may use information retrieval techniques similar to the ones in [17].

3.1.2 Range Descriptions

Once the super-peer has identified the set Ps it must decide, based on the range conditions in Q, which peers from Ps have data relevant to the query. In order to identify the relevant peers, the super-peer keeps range information for the neighbouring peers in its catalogue. For a peer P, range information for an attribute r that is functionally dependent on a set of attributes $K = \{k_1, k_2, ..., k_n\}$, is specified by a set R of tuples:

$$R = \{ \quad (v_{11}, ..., v_{n1}, min_1, max_1, nvals_1), \\ (v_{12}, ..., v_{n2}, min_2, max_2, nvals_2), \\ ..., \\ (v_{1m}, ..., v_{nm}, min_m, max_m, nvals_m) \quad \}$$

where v_{ji} represents a valuation of attribute k_j in the $i - th$ tuple; $(min_i, max_i, nvals_i)$ represent the minimum, maximum, and number of values for attribute r when the attributes $k_1, ..., k_n$ have values $v_{1i}, ..., v_{ni}$. For the example in Figure 1, if $r = product\_cost$ and $K = \{product\_class\}$, a tuple specifying range information for TVs will look like $(TV, (100, 300, 20))$. This tuple indicates that there are 20 TVs in the \$100 and \$300 price range. A more detailed range specification would be $K = \{product\_class$, $product\_unit$, $product\_dim\}$. In this case, an example tuple representing the fact that there are 10 TVs of 17 inches in the price range \$100 and \$150, would be $(TV, inches, 17, 100, 150, 10)$. A detailed range specification can be used only if queries specify values for all attributes in K. Hence the type of queries expected to be executed should influence the specification of ranges.

The level of detail when specifying ranges must strike a balance between the number of tuples in R and how discriminatory the range is. If R is too large, storage costs increase; if R is too small, irrelevant queries may be sent to a peer. In this latter case, the effect would be similar to that of not having range-specific information to optimise range queries. In general, the number of values in all ranges should be much greater than the number of entries in R:

$$\sum_{0 \leq i \leq m} nvals_i \gg |R|$$

Another important consideration is the distribution of values within a range. In a range specification $(min_i$, max_i, $nvals_i)$, we assume a uniform distribution of values between min_i and max_i. Hence, if $(min_i, max_i, nvals_i) = (100, 500, 20)$, and a query specifies a range condition with values 400 and 500, we assume that the peer will potentially have five values in the 400 to 500 range. The number of values in a range is used by the super-peer to rank peers relevant to a query. Assuming a range condition $q_{min} \leq r \leq q_{max}$, the ranking of peers is determined by the sum of the expected number of values in the ranges intersecting the query range:

$$\sum_{\substack{i: q_{min} \leq max_i \wedge \\ q_{max} \geq min_i}} nvals_i * \frac{\min(q_{max}, max_i) - \max(q_{min}, min_i)}{max_i - min_i}$$

This rank is used by the super-peer to decide the order in which the query is sent to peers for execution. Favouring peers with more data within a query range may or may not be desirable based on the type of application running in the system. For example, the ranking is acceptable when integrating bibliographic databases, where users may want to receive as many answers as possible. On the other hand, in a commerce application integrating stores selling goods, users may be more interested in price than volume. Similarly, the system may need to guarantee that no store is favoured over others. Ranking of candidate peers to resolve a query is nevertheless an interesting problem. When the number of peers that have relevant data to a query is potentially very high, sending the query to all peers for execution is not a viable solution. Besides the ranking, other options to further limit the number of peers participating in a query include imposing a ceiling based on a system parameter or specified as part of the query, and to only consider answers received within a time interval. Yet another option, is to distribute the load so that queries are sent to relevant peers in a round-robin fashion. Therefore, due to the potential diversity of applications, a P2P system should provide several configurable options to allow tuning and enhancement of the techniques used to select among the peers with relevant data.

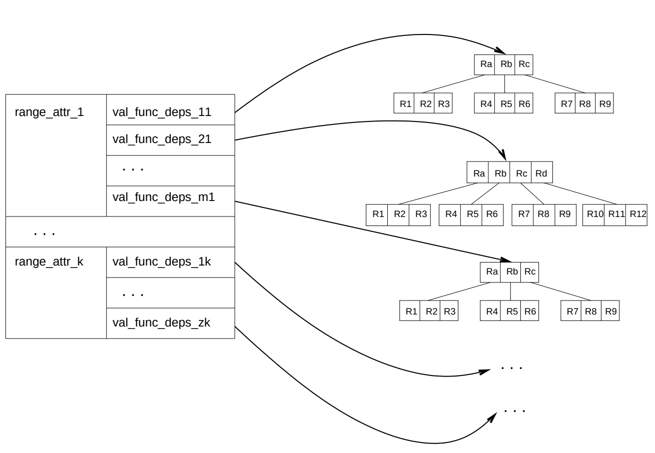

Figure 3: SP-to-P Catalogue Structure

Range information is provided by the peers to the neighbouring super-peer as part of the registration of their data sources. Logically, the data source registration is stored in the SP-to-P catalogue as shown in Figure 3. Associated to each range attribute there are sets of valuations for the attributes in K (recall that K is the set of attributes a range attribute depends on). An R-tree [13, 6] indexes the ranges for the attribute given a K valuation. The leaves in the R-tree point to the peers that store data in the given range and the number of values in the range for each peer.

When a peer specifies the set R that describes the range information for an attribute r, the super-peer integrates this information to the catalogue by following the Algorithm 1. Range information is added to the R-tree associated to the valuation of the attributes in K. If no entry in the catalogue matches the range attribute r and the valuation of attributes in K, a new entry is added to the catalogue. This new entry is associated to a new R-tree containing the given range. Since the names of the attributes in R may not match the names in the catalogue, the super-peer uses the synonyms specified in the range registration to determine if an entry for a $r, v_{1i}, ..., v_{ni} \in R$ already exists in the catalogue or not.

Algorithm 1: Addition of Range Information

```
1 foreach $(v_{1i}, ..., v_{ni}, min_i, max_i, nvals_i) \in R$ do
2     if $\exists val\_func\_deps : (r, val\_func\_deps) \in catalogue \wedge val\_func\_deps = (v_{1i}, ..., v_{ni})$ then
3         Add the range $(min_i, max_i)$ to the R-tree
          associated to the catalogue entry for
          $(r, val\_func\_deps)$
4     else
5         create a new catalogue entry $(r, v_{1i}, ..., v_{ni})$
          and a R-tree with the range $(min_i, max_i)$
6     end
7 end
```

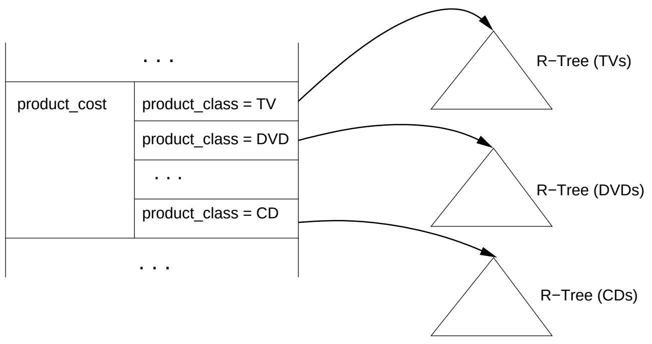

Figure 4: SP-to-P Catalogue Example

To illustrate Algorithm 1, Figure 4, shows the catalogue entry for the attribute *product_cost*, assuming that this attribute is functionally dependent on the attribute *product_class*. A peer specifies a set R of ranges for *product_cost*, and values for *product_class* as follows:

$$R = \{ \quad (TV, 150, 3500, 80), \\ (DVD, 2, 60, 1300), \\ (Printer, 40, 300, 25) \quad \}$$

For $(TV$, 150, 3500, 80), the algorithm adds the range (150, 3500) to the R-tree associated with *product_class* $= TV$. The node in the tree corresponding to the range points to the peer's identity and the the number of values in the range (80). Since there is an entry for *product_class* $= DVD$, similar processing occurs for $(DVD$, 2, 60, 1300). Assuming no entry in the catalogue for *product_class* $=$ *Printer*, processing $(Printer$, 40, 300, 25) requires the creation of a new entry for *product_cost* with valuation *product_class* $=$ *Printer*, and the association of a new R-tree containing the range (40, 300).

3.1.3 Scalability Considerations

The proposed catalogue structure assumes high similarity, not only in the attribute and entities among data sources, but also in the functional dependencies for the range attributes. If the functional dependencies differ considerably between peers, the proposed catalogue does not scale well. The reason is the potential for having as many entries and R-trees as different valuations of functional dependencies exists. In this situation, it is better to alter the structure of the catalogue and associate the R-trees to the range attributes directly. In this variation, nodes of the R-trees point to the peer identifiers, the number of values in the range, and the valuation of dependent attributes.

Once a super-peer has identified and ranked the neighbouring peers storing data relevant to a query, the next step in the processing of the query is the routing of the query towards super-peers close to promising peers. The selection of these super-peers is based on the information contained in the SP-to-SP part of the catalogue.

3.2 SP-to-SP Catalogue Information

Super-Peers summarise their own SP-to-P catalogue and make the summary available to other super-peers. The summaries from remote super-peers constitute the SP-to-SP catalogue.

3.2.1 Schema Summaries

Information about the entities and attributes occurring at neighbouring peers is summarised by aggregating the synonyms and eliminating duplicated names. The aggregation of synonyms follows the Algorithm 2. For an entity E_p in the SP-to-P catalogue, the condition in line 3 identifies the entity E_s already in the summary that has similar name and attributes to E_p. The function *summ_syns* in the algorithm is equivalent to the function *syns* defined for the SP-to-P catalogue but refers to the synonyms for a particular entity or attribute in the SP-to-SP catalogue instead. If such entity E_s exists in the summary, the synonyms of E_p not already in the synonyms of E_s are added in line 4. Similar processing for the attributes of E_p is done in lines 5 to 7. If there is no entity E_s in the summary that corresponds to E_p, E_p is added to the summary in lines 9 and 10, and E_p's attributes are added in the loop in lines 11 to 13. The intention of the algorithm is to identify if an entity in one peer is already in the summary, and only add new entries otherwise.

Algorithm 2: Aggregation of Entity/Attribute Information

```
1  $summ = \{\}$
2  foreach entity $E_p$ in SP-to-P do
3      if $\exists E_s \in summ$ :
       $summ\_syns(E_s) \cap syns(E_p) \neq \emptyset \wedge$
       $(\forall a_s \in attrs(E_s) : \exists a_p \in attrs(E_p)$ :
       $summ\_syns(E_s, a_s) \cap syns(E_p, a_p) \neq \emptyset)$ then
4          $summ\_syns(E_s) =$
           $summ\_syns(E_s) \cup syns(E_p)$
5          foreach $a_s \in attrs(E_s)$ do
6              $summ\_syns(E_s, a_s) =$
               $summ\_syns(E_s, a_s) \cup syns(E_p, a_p)$
               where $a_p$ is the attribute in $E_p$ such that
               $summ\_syns(E_s, a_s) \cap syns(E_p, a_p) \neq \emptyset$
7          end
8      else
9          $summ = summ \cup E_p$
10         $summ\_syns(E_p) = syns(E_p)$
11         foreach $a_p \in attrs(E_p)$ do
12             $summ\_syns(E_p, a_p) = syns(E_p, a_p)$
13         end
14     end
15 end
```

Summaries are distributed to other super-peers. Super-Peers receiving the summary, tag the information with the identifier of the source super-peer. These summaries allow a super-peer S to identify a set SPs of super-peers close to data sources with entities and attributes relevant to a query Q. As with the SP-to-P catalogue, the identification of super-peers SPs is based on the existance of a function *sim* able to decide if two entities or attributes are similar.

3.2.2 Range Summaries

Given a range attribute r and a valuation of the dependency attributes in the SP-to-P catalogue, the range information for the attribute r is summarised by taking the minimum and maximum values in the R-tree for the attribute. The number of values associated to r in the summary is computed by adding the number of values for each of the ranges in the R-tree. A super-peer stores the summaries for each other super-peer in a structure similar to the one used to store ranges in the SP-to-P catalogue (see Figure 3). The main difference between the SP-to-P representation of ranges and the SP-to-SP representation is that the R-trees associated to the ranges in the SP-to-SP catalogue refer to summary ranges for particular super-peers instead of peers. Hence the leaves of the R-tree in the SP-to-SP catalogue point to super-peers and their number of values for the specific range. The number of values is used to rank super-peers relevant to a query. Considerations about this ranking and its applicability are similar to the ones presented in section 3.1 for peers.

3.2.3 Maintenance of Summaries

Associated to each super-peer summary there is a monotonically increasing version number and a TTL specified by the super-peer providing the summary. A super-peer sets the TTL based on the frequency of SP-to-P registration updates made by its peers. Summaries can be sent (*pushed*) to the super-peers when the TTL expires, or super-peers can request the summary (*pulled*) if the local copy has expired. The version number of the local copy at the super-peer is transmitted to the source super-peer as part of the request for the summary. The source super-peer can then provide the new summary if the version is out of date, or inform the super-peer if the version is still the latest. Whether to implement pushed or pulled distribution of summaries depends on the expected number of super-peers in the system and the characteristics of the applications running on the system. Pushing summaries may be acceptable when the number of super-peers in the system is small, or the summaries are not frequently updated.

As with entity and attribute information, the space required to store the range information in the SP-to-SP catalogue is proportional to the semantic similarity of the data sources close to each of the super-peers. Another factor affecting the space requirements is the number of super-peers in the system. In cases where the number of super-peers or the semantic disparity among data sources impose unacceptable space requirements for the SP-to-SP catalogue, an alternative to storing the summary of each super-peer is to request the summary information on-demand. In this scenario, super-peers would maintain basic schema information about the other super-peers in the system. For each other super-peer, the SP-to-SP catalogue would store just the name and synonyms of the entities in data sources close to the super-peer. When processing a query Q in a super-peer S, the SP-to-SP catalogue would be used to determine the super-peers that may potentially be relevant to Q. S would then contact each of the identified super-peers and request the summary for the attributes and ranges in Q. This information would then be used to identify the super-peers close to data sources storing data in the data ranges specified in Q.

To improve performance of the query processing, and assuming that there are some queries more popular than others, summary information can be cached locally at super-peers when requested on-demand. The version information associated to each summary can be used to determine if a cached summary is still valid once the TTL associated to the summary has expired.

3.3 Propagation of Peer Registrations

When peers re-register their data sources, changes with respect to the previous registration need to be propagated to the SP-to-P and SP-to-SP catalogues. A naive approach to implement the propagation is to delete the previous registration, and add the new one. This approach does not work well when the data re-registered by the peer is similar to its previous registration. A more efficient approach is to compare the new and previous registration and propagate only the changes. Possible changes are structural modifications and range changes. Structural modifications include the addition or removal of entities or attributes. Range changes include the modification of existent ranges, and the specification of new ranges and valuations for the dependency attributes. The propagation of the change to the SP-to-P catalogue is implemented by actions local to the super-peer. The propagation of the change to the SP-to-SP catalogue needs to be handled with care since it affects all the copies of the super-peer's summary in the system. A super-peer may decide to propagate changes to its summary information only after a number of changes threshold is exceeded, after a time interval, depending on the change, or any combination of these. Irrespective of the strategy used, when the summary is updated, the version number associated to the summary should be increased. In order to reduce the amount of data transmitted to other super-peers when a summary changes, the super-peer could send the delta between the old and new versions of the summary. Deltas would then be applied to the local copy of the summary by the receiving super-peers.

4. LOAD DISTRIBUTION STRATEGIES

Load distribution imbalances may occur in the system when some ranges are more popular than others. Another reason is the existence of few data sources relevant to a popular query. Given our assumption that the location of the data sources cannot be altered, strategies to balance the load distribution in the system need to be implemented at the super-peer level.

When a range is popular, the super-peers close to the data sources relevant to the range will receive requests for summary information and execution of the queries. By implementing caching of summary information as proposed in Section 3.2, the system reduces the load on the popular super-peers. Some strategies to reduce the load due to requests for query execution include:

- Splitting the range among several super-peers
- Caching query results at the super-peer where the query originated
- Reassigning peers with data sources to other less loaded super-peers
- Combinations of the above

In a range split, a range g is divided into sub-ranges g_1, g_2, ..., g_n, such that $g = g_1 \cup g_2 \cup ... \cup g_n$ and $g_i \cap g_j = \emptyset$ for $i \neq j$. Ideally a data source for a peer P would have data relevant to only one g_i. If this is not the case, the peer P will be associated to all super-peers hosting ranges relevant to P's data. The range re-registration process at P needs to be aware of what ranges are hosted in what super-peers, and only re-register applicable information with each of these super-peer. This procedure increases the state information kept at P. An alternative is to have P re-register all of its information with the original super-peer, and have this super-peer split the registration information and send the pieces to the super-peers hosting the sub-ranges. In this case, the re-registration functionality in P is unaffected by the range split. The state information in the super-peer is increased though, since it now needs to keep track of the ranges that have been split. The original super-peer still needs to use some of its resources to process P's re-registrations. Queries arriving at the super-peers hosting the sub-ranges are re-routed to the original super-peer, unless direct communication to P is allowed from super-peers P is not attached to.

Caching query results at super-peers is an effective way to balance the load when changes to the data sources are infrequent. Maintenance of the cached copies becomes an issue when the data source are updated or when there is high peer volatility in the system.

Reassigning peers to other data sources to distribute the load is probably the most straightforward way to deal with load balances in the system. Super-peers need to keep information about its load and be able to rank the neighbouring peers based on how much load they contribute to the super-peer. When the load at a super-peer exceeds a threshold, the super-peer identifies what other, less loaded super-peers, can take some of its neighbouring peers. The load threshold can be individually set by each super-peer based on its capacity, or the system can impose the same value for all super-peers.

As with other functionality proposed in this work, a P2P system should implement various techniques for load balancing and allow the configuration and tuning of these techniques based on specific application requirements.

5. IMPLEMENTATION AND EXPERIMENTATION

In this section we present experimental results obtained by implementing some of the techniques proposed in our catalogue framework using the JXTA P2P platform [1, 22]. We describe the implementation details in the context of JXTA and report the results obtained by conducting a set of experiments evaluating different performance metrics.

5.1 Implementation in JXTA

JXTA is a set of protocol specifications, developed by Sun Microsystems, to handle the communication among participants in a P2P system. The JXTA protocols are language-independent, and define a set of XML messages to coordinate different aspects of P2P networking.

Peers publish the functionality they offer to other peers as *services* in the network. These services are available only when the peer is connected to the system. Peers use the JXTA *Peer Discovery Protocol* to discover the services that other peers have published. One of the services that can be offered by a peer is its readiness to accept direct peer connections. The JXTA *Pipe Binding Protocol* provides a mechanism to bind peers through pipes, which are virtual communication channels that encapsulate all networking details. A Pipe is bound to a certain TCP port at one peer and can be discovered by other peers when it is published as a *service* in the JXTA network. Pipes are *unidirectional*, meaning that data travels in only one direction. However, *bidirectional* pipes can be implemented using two *unidirectional* pipes. Pipes are also *asynchronous*, meaning that data can be sent or received at any time, which allows peers to act independently of one another.

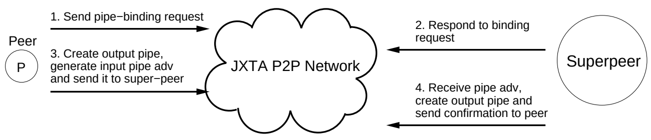
1. Send pipe−binding request
Peer
P
3. Create output pipe, generate input pipe adv and send it to super−peer
JXTA P2P Network
2. Respond to binding request
4. Receive pipe adv, create output pipe and send confirmation to peer
Superpeer

Figure 5: Establishing Bidirectional Pipes

5.1.1 Peer Registration

In our implementation, a super-peer declares its readiness to serve other peers by creating an input pipe on a free TCP port and publishing the pipe advertisement, which is an XML file incorporating the pipe ID and other related information. A peer, that chooses to connect to that super-peer, uses the *Discovery Protocol* to locate the super-peer pipe advertisement. The peer sends a discovery message to other known peers which inspect their own caches for the required advertisement and either respond to the discovery request or forward the request to other peers. Eventually, information about pipe advertisement is located and sent back to the initiating peer. At that time, the peer sends a pipe-binding request to the super-peer which responds with a pipe-binding answer message confirming that the binding request is accepted. When the peer receives the answer message, it creates an output pipe to bind itself to the super-peer. Furthermore, the peer creates an input pipe and sends its advertisement (using its output pipe) to the super-peer to be able to receive further messages from the super-peer in an asynchronous fashion. Therefore, a *bidirectional* pipe is established between the peer and super-peer for future message exchange. The pipe setup procedure is depicted in Figure 5.

The super-peer's input pipe serves as a common entry point that is shared among all the connected peers and is used to submit peer requests to the super-peer. On the other hand, peer input pipes are specific to the owning peer; i.e. a peer input pipe receives messages from the super-peer directed only to that peer. The super-peer keeps a list of all the input pipe IDs of connected peers. The peer attaches its input pipe ID to every message sent to the super-peer so that the super-peer can identify the source of the message and therefore return the answer via the correct path.

The first message sent by the peer to the super-peer, after establishing the communication path, describes the data the peer is willing to share with other peers. As specified in Section 3.1, this is referred to as *peer registration*. In our implementation, a peer registration message is an XML file describing a certain *property*, e.g. price, its range, number of values and a set of keywords related to that property. The property corresponds to a range attribute. The set of keywords specify the synonyms or alternate names for the property. Keywords are used by the super-peer to cluster the property registrations received from "schema-heterogeneous" peers into separate groups each containing potentially similar range attributes. This is based on the intuition that the more common keywords between two properties, the more likely it is that they are semantically close.

```
<?xml version="1.1"?>
<PeerRegistration>
    <PipeId>
        urn:jxta:uuid-5961626164162614E50472050325033222
        CD113577B4B29BEAFD52FCC2CD2D204
    </PipeId>
    <Property>
        <Name> Price </Name>
        <Range> 100 200 </Range>
        <NumValues> 40 </NumValues>
        <Keywords>
            Television, TV, Cost, Charge, ProductPrice
        </Keywords>
    </Property>
    <TTL>
        <Hours> 2 </Hours>
        <Minutes> 30 </Minutes>
    </TTL>
    <Attributes>
        <Attribute>
            <Name> ProductClass </Name>
            <Value> TV, Television </Value>
            <Keywords>
                class, type, prod_class, category, prod_category
            </Keywords>
        </Attribute>
        <Attribute>
            <Name> ProductSize </Name>
            <Value> 20 </Value>
            <Keywords>
                size, dim, dimension, screen_size, screen_dim
            </Keywords>
        </Attribute>
    </Attributes>
</PeerRegistration>
```

Figure 6: Peer Registration Message

The peer *registration* message also lists some of the attributes that *property* is functionally dependent on. Each of these attributes is described by its name, value and keywords. The super-peer stores peer registrations for some predefined TTL period as indicated by the message. The peer has the responsibility of refreshing its registration at the super-peer before the TTL expires, otherwise it will be dropped from the SP-to-P catalogue. Figure 6 illustrates a sample *registration* message describing the price property of TV products that a peer is selling.

5.1.2 Catalogue Structure

The registration messages sent by different peers are used to build the SP-to-P catalogue at the super-peer. The catalogue groups peers according to their data attributes and range information. In addition, each super-peer keeps track of all the other known super-peers in the SP-to-SP catalogue.

The main function of the SP-to-P catalogue is to support the identification of promising peers that can answer a given range query. These promising peers can be determined, with a high probability, by inspecting the following peer properties:

- **Keywords:** The number of common keywords between the peer's data source and the query.
- **Range Information:** Whether the range in the peer's data source intersects with the range specified by the query or not.

Peers are filtered based on keywords first, and then based on range information.

The identification of ranges that intersect with the query range is performed by using R-trees. We use 1-dimensional R-trees to index range information of different sets of peers. Given a certain query range, traversing the R-tree allows to efficiently return the set of peers with ranges intersecting with the query range.

```
<?xml version="1.1"?>
<PeerQuery>
    <PipeID>
        urn:jxta:uuid-5961626164162614E50472050325033222
        CD113577B4B29BEAFD52FCC2CD2D204
    </PipeID>
    <Property>
        <Name> Cost </Name>
        <Range> 50 150 </Range>
        <Keywords>
            TV, Price, TVPrice ,TVCost, Television
        </Keywords>
    </Property>
    <Attributes>
        <Attribute>
            <Name> Type </Name>
            <Value> TV, Television </Value>
            <Keywords>
                class, category
            </Keywords>
        </Attribute>
    </Attributes>
</ PeerQuery >
```

Figure 7: Peer Query Message

The SP-to-P catalogue is comprised of a set of n entries $\{< k_i, R_i >; i=1..n\}$; where k_i is the keyword vector for the i^{th} peer set and R_i is an R-tree that indexes the range information of all peers inside the i^{th} peer set. When a peer registration message is received by a super-peer, the peer's property name and keywords are extracted forming a peer keyword vector v. The super-peer finds every keyword vector k_j where $\frac{|v \bigcap k_j|}{|v|} \geq \texttt{MIN\_SIMILARITY}$. The peer keyword vector v is then merged with all k_j's and the peer range information is inserted in all R_j's. If the set $\{k_j\}$ is empty, a new entry is appended to the SP-to-P catalogue containing v and a R-tree indexing only the peer range information. `MIN_SIMILARITY` is a system parameter that was decided to be 0.4 after conducting experiments. Hence the *similarity* function (Section 3.1.1) in our implementation is defined based on the number of matches among keyword vectors and a threshold `MIN_SIMILARITY` value.

The SP-to-SP catalogue stores summary information about the neighbouring super-peers. Each super-peer sends periodically a summarised registration message to all the known super-peers containing information about the peers connected to it. Hence, summary information for the SP-to-SP catalogue is registered by super-peers following the *pushing* strategy described in Section 3.2.3. The super-peer registration message contains a set of keyword vectors describing different peer properties. The super-peer registration expires after the TTL period specified in its registration. The SP-to-SP catalogue is comprised of a set of m entries $\{< k_i, sp_i >; i=1..m\}$; where k_i is the keyword vector associated with the super-peer set sp_i. The catalogue entries are constructed based on keyword intersection in a way similar to the SP-to-P catalogue described above.

5.1.3 Query Processing

Queries are represented as XML messages submitted by peers to super-peers. The query message specifies the required data property, range, property keywords and additional data attributes. The keywords are used to identify peers that support that property under potential naming or structural heterogeneity. A sample query message is shown in Figure 7.

The super-peer uses its catalogue to forward the received query to potentially promising peers. The SP-to-P catalogue is propped with the query keyword vector q that contains property name and keywords. The super-peer finds all entries in the SP-to-P catalogue with a keyword vector k_j satisfying $\frac{|q \bigcap k_j|}{|q|} \geq \texttt{MIN\_SIMILARITY}$. The R-trees for these entries are used to identify the peers with data intersecting the query range. The registration messages for those peers are then loaded from the super-peer catalogue. The additional attributes included in peer registration messages are compared against the additional query attributes to find the potentially equivalent attribute pairs based on common keywords. A peer is considered a candidate to answer a query if its attributes are equivalent to at least 60% of the query attributes.

The resulting promising peers are ranked based on the expected number of relevant results that they can return. This rank is based on the assumption that data values are uniformly distributed over data ranges, as described in Section 3.1.2. Peers are then queried in rank order after reformulating the queries according to the registration of each peer. The results are sent back to the super-peer which forwards any received results to the peer who initiated the query in a streaming fashion.

The SP-to-SP catalogue is also propped using a keyword vector containing all query keywords. The common keywords between query and SP-to-SP catalogue entries are used to decide the super-peers that a query should be forwarded to. Results are sent back to the super-peer where the query initiated.

5.2 Experimental Results

To study the performance of the proposed techniques, we used J2SE5.0 to implement the discussed functionality on the JXTA platform. All the experiments were run on a 3GHz Pentium IV PC with 1 GB of RAM and a 40 GB hard disk, running Windows XP.

We spawned multiple instances of peers and super-peers on the same PC to represent an actual P2P network. Peer instances bootstrap to the JXTA network in an uncontrolled manner. This means that different peer instances could be connected to the JXTA network through different intermediate peers. Therefore, messages of one peer instance have to pass through the JXTA network to reach other peer instances on the same PC. Furthermore, JXTA protocols do not guarantee message delivery. A message can be lost if it is passed to an intermediate overloaded peer. We believe that these conditions simulate actual P2P environments.

The keyword datasets were collected from commercial web sites of *TV*, *stereo* and *car* vendors. The collected keywords include around 30 different alternate names for each one of the most commonly used data attributes of these products. Product *price* was taken as the property for which range conditions are most frequently specified in queries. Peer registration messages were generated by selecting keywords from the datasets independently and uniformly at random. Each registration message describes a certain product with a random number of product attributes besides price. Each data attribute is associated with a maximum of 5 different keywords to represent the case where users do not associate many keywords in their product registrations.

The system setup is comprised of a P2P network with 3 super-peers serving a number of peers ranging from 50 to 3000. Each peer selects its super-peer randomly and registers only one product type. A query message requests data about a certain product type (e.g TVs) by specifying randomly selected price ranges and data attributes. The query attributes are associated with keywords selected from the keyword datasets of that product.

5.2.1 Recall and Precision

In this experiment we measure the *Recall* and *Precision* metrics for query results. We use the standard definitions of both metrics:

$$Recall = \frac{No.\ of\ relevant\ retrieved\ results}{Total\ no.\ of\ relevant\ results}$$

$$Precision = \frac{No.\ of\ relevant\ retrieved\ results}{Total\ no.\ of\ retrieved\ results}$$

A data source (peer) is defined to be *relevant* to the query if its registration message: (i) is about the product requested by the query, (ii) specifies a range that intersects with the query range, and (iii) includes all the attributes specified by the query. Any registered peer with the previous properties is expected to return answers for the query.

We injected different queries into the network and compared the specifications of each query message against the registration messages of the returned peers as well as the rest of peers existing in the network. Figures 8 and 9 show the measured Recall and Precision metrics of query answers obtained for different product types. Each point is the average of 5 independent runs with a different query message in each run.

The recall metric shows stable system behaviour regardless of the network size. The standard deviation for the recall metric is around 0.01 for each of the three product types. The relatively small recall levels can be attributed to the fact that all query messages had a maximum of only 5 keywords associated to each of their attributes. The recall level depends on the number of keywords associated with query attributes. By increasing the number of keywords, the size of the returned results also increases. This is evident by repeating the same experiments with a maximum of 10 keywords per each query attribute. The Recall level increased by an average of 35% ,22% and 16% for TV, stereo and car queries respectively.

On the other hand, the precision metric dropped by an average of 19% when increasing the network size from 50 to 3000 peers. The drop is expected because by increasing the network size, the number of registrations with different keyword specifications also increases, which makes it harder to find relevant results using queries of limited number of keywords.

It can be inferred from these results that using super-peers to give feedback to the peers about the mostly used attribute keywords in peer registrations can enhance both the recall and precision. Queries expressed in widely used keywords are expected to find more matches than the ones expressed in randomly chosen keywords.

5.2.2 Response Time

In this experiment we measure the average query response time for different network sizes. The query response time is affected by the number of peers that are contacted to process the query. Reducing this number of participant peers is crucial in P2P systems to guarantee their scalability. The SP-to-P and SP-to-SP catalogue information are primarily

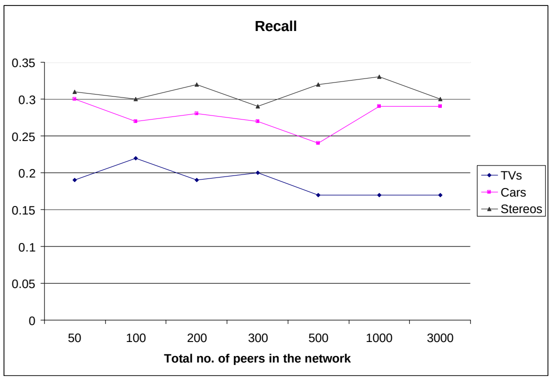

Figure 8: Recall

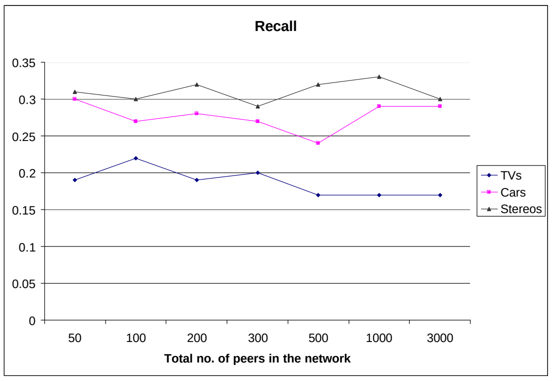

Figure 9: Precision

used to for this purpose in order to limit flooding queries to only potentially relevant peers.

To study the effect of increasing the network size on the query response time, we injected the same queries into networks with different number of peers and measured the average response time of 5 different queries in independent runs over each network size. Table 1 shows the results obtained for different query types.

The average response time increased by a factor of 6 while the network size increased by factor of 600. The measured response time represents the actual communication time plus the time taken by super-peers to process queries, contact relevant peers, and aggregate results. The measured response time did not exceed 0.13 seconds for the largest tested network (3000 peers); this result shows the efficiency of the proposed catalogue services in pruning the search space.

5.2.3 *Effect of Range Information*

The purpose of this experiment is to evaluate the effect that the range information in the catalogue has on recall and precision. In order to carry out this experiment we created a variation of our system where the range information (indexed by R-trees) is omitted from the catalogue. Hence in this variation, peers relevant to a query are selected based on structural information (data attributes) only. Since this is the default behaviour of most unstructured P2P systems, this experiment provides some insight as to how our techniques improve processing of range queries when compared with unoptimised systems.

The recall was not affected by the omission of range information. This is due to the fact that the set of peers found relevant to the queries is a super set of the set of peers found relevant when using range information. On the other hand, precision dropped significantly due to the increase in the number of peers to which the query is forwarded. Figure 10 shows the precision drop experienced when omitting range information from catalogues as the number of participants in the system increases. The average precision drop is around 49% over three different query types and different network sizes. The result emphasises the importance of indexing data ranges to support efficient range queries in unstructured P2P environments when the schemas are semantically close.

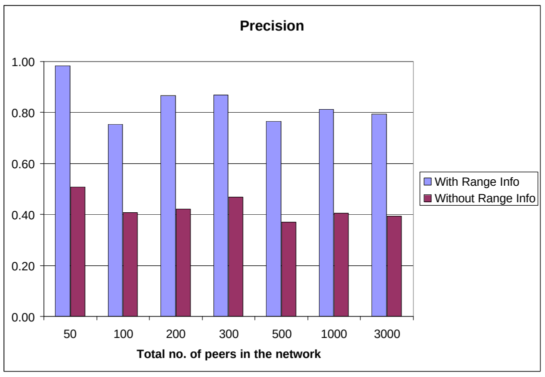

Figure 10: Effect of Range Information on Precision

Table 1: Average Query Response Time (msec)

Network Size	TV Queries	Car Queries	Stereo Queries
50 peers	18	15	16
100 peers	26	16	19
300 peers	29	22	22
500 peers	39	25	29
1000 peers	66	33	47
3000 peers	127	85	124

6. CONCLUSIONS AND FUTURE WORK

Research on data sharing P2P systems should focus on analysing the requirements imposed by the applications running on such systems. In this work we have looked at range queries and the type of functionality required to efficiently support their processing in unstructured P2P systems. Specifically, we presented a catalogue framework that allows the identification of peers with data sources relevant to a given query. The framework is based on the assumption of high semantic similarity among the data sources. We have looked at different types of applications and how the framework can be adjusted to meet their requirements. Through experimentation we have study the performance of our catalogue, and we have found a substantial improvement in precision when evaluated against a P2P system implementing the traditional attribute-occurrence approach for selection of peers relevant to a query.

Future work related to our proposal includes the evaluation of the different alternatives presented for the implementation of the catalogue. Areas worth investigating are the efficient representation of summaries in SP-to-SP catalogues, the distribution of changes in the catalogue, techniques for ranking peers relevant to a query to avoid starvation or unfair situations, elimination of the assumption of uniform distribution of data ranges and possibly the use of histograms instead, evaluation of load balancing techniques, and development of a P2P system with open interfaces to allow for the experimentation and evaluation of diverse implementation alternatives.

7. REFERENCES

[1] Project jxta web site: http://www.jxta.org.

[2] K. Aberer, P. Cudr-Mauroux, and M. Hauswirth. The chatty web: Emergent semantics through gossiping. In *WWW '03: Proceedings of the twelfth international conference on World Wide Web*, pages 197–206. ACM Press, 2003.

[3] A. Andrzejak and Z. Xu. Scalable, Efficient Range Queries for Grid Information Services. In *Proceedings of the IEEE International Conference on P2P computing*, 2002.

[4] J. Aspnes, J. Kirsch, and A. Krishnamurthy. Load balancing and locality in range-queriable data structures. In *Proceedings of the 23rd annual ACM symposyum on Principles of Distributed Computing, PODC 04*, pages 115–124, 2004.

[5] J. Aspnes and G. Shah. Skip graphs. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 384–393, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[6] N. Beckmann, H.-P. begel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD Conference Proceedings*, pages 322–331, 1990.

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

[8] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. volume 34, pages 353–366, New York, NY, USA, 2004. ACM Press.

[9] M. Cai, M. Frank, J. Chen, and P. Szekely. Maan: A multi-attribute addressable network for grid information services. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 184, Washington, DC, USA, 2003. IEEE Computer Society.

[10] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating data sources in large distributed systems. In *Proceedings of the 29th VLDB Conference*, 2003.

[11] Gnutella. `http://rfc-gnutella.sourceforge.net/`, 2005.

[12] A. Gupta, D. Agrawal, and A. Abbadi. Approximate Range Selection Queries in Peer-to-Peer Systems. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.

[13] A. Guttman. R-trees : A Dynamic Index Structure for Spatial Searching. In *Proc. of ACM SIGMOD Int. Conf. on the Management of Data - SIGMOD Record 14(2)*, pages 45–57, 1984.

[14] KaZaA. `http://www.kazaa.com/`, 2005.

[15] Napster. `http://www.napster.com/`, 2001.

[16] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: a p2p networking infrastructure based on rdf. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 604–615, New York, NY, USA, 2002. ACM Press.

[17] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. In *Intl. Conf. on Data Engineering (ICDE)*, 2003.

[18] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.

[19] O. Sahin, A. Gupta, D. Agrawal, and A. Abbadi. Query Processing Over Peer-to-Peer Data Sharing Systems. Technical report, University of California at Santa Barbara, 2002.

[20] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.

[21] I. Tatarinov, Z. Ives, J. amd, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyaska, G. Miklau, and P. Mork. The piazza peer data management project. *ACM SIGMOD Record*, 32(3), 2003.

[22] B. J. Wilson. *JXTA. A Text Book , http://www.brendonwilson.com/projects/jxta/index.shtml*. 2001.