Selective Information Dissemination in P2P Networks: Problems and Solutions*

Manolis Koubarakis Christos Tryfonopoulos Stratos Idreos Yannis Drougas

Intelligent Systems Laboratory
Dept. of Electronic and Computer Engineering
Technical University of Crete
GR73100 Chania, Crete, Greece
http://www.intelligence.tuc.gr

ABSTRACT

We study the problem of selective dissemination of information in P2P networks. We present our work on data models and languages for textual information dissemination and discuss a relevant P2P architecture that motivates our efforts. We also survey our results on the computational complexity of three related algorithmic problems (query satisfiability, entailment and filtering) and present efficient algorithms for the most crucial of these problems (filtering). Finally, we discuss the features of P2P-DIET, a super-peer system we have implemented at the Technical University of Crete, that realizes our vision and is able to support both ad-hoc querying and selective information dissemination scenarios in a P2P framework.

1. INTRODUCTION

In peer-to-peer (P2P) systems a very large number of autonomous computing nodes (the *peers*) pool together their resources and rely on each other for *data* and *services*. P2P systems are application level *virtual* or *overlay networks* that have emerged as a natural way to share data and resources. Popular P2P *data sharing* systems such as Napster, Gnutella, Freenet, KazaA, Morpheus and others have made this model of interaction popular.

The main application scenario considered in recent P2P data sharing systems is that of *ad-hoc querying*: a user poses a query (e.g., "I want MP3s with Jennifer Lopez") and the system returns a list of pointers to matching files owned by various peers in the network. Then, the user can go ahead and download files of interest. The complementary scenario of *selective information dissemination (SDI)* or *selective information push* [11] has so far been considered by very few P2P systems [5, 18, 23, 12]. In an SDI scenario, a user posts a *profile* or *continuous query* to the system to receive notifications whenever certain *events* of interest take place (e.g., when a video-clip of Jennifer Lopez becomes available). SDI can be as useful as ad-hoc querying in many target applications of P2P networks ranging from file sharing, to more advanced applications such as alert systems for digital libraries, e-commerce networks etc.

*This work was carried out as part of the DIET project (IST-1999-10088), within the UIE initiative of the IST Programme of the European Commission.

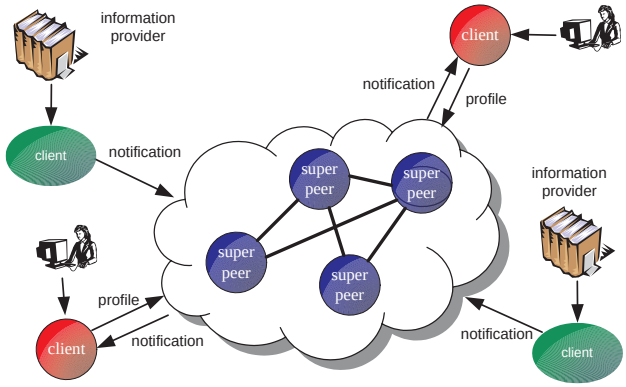

Figure 1: A P2P architecture for SDI

At the Intelligent Systems Laboratory of the Technical University of Crete, we have recenlty concentrated on the problem of SDI in P2P networks. In this paper we survey our work in this area, summarize our main results and compare with related research. For more details the interested reader should consult the papers [17, 20, 18, 19] and technical reports [27, 14].

The rest of the paper is organized as follows. Section 2 presents our favourite P2P network architecture for SDI. Then Section 3 summarizes our contributions on related data models, query languages, computational complexity analysis and efficient filtering algorithms. Section 4 discusses the system P2P-DIET. Finally, Section 5 presents our conclusions.

2. AN SDI ARCHITECTURE BASED ON SUPER-PEERS

We have studied SDI scenarios in the context of the P2P architecture of Figure 1 which is based on *super-peers*. Super-peer architectures for P2P data sharing networks have recently been analyzed in [30] for the standard ad-hoc querying scenario. The architecture of Figure 1 was originally proposed in [5] in the context of the *distributed event notification system* SIENA and later-on adopted by us in [18] (SIENA uses the term *server* instead of super-peer but the

idea is exactly the same).

In the architecture of Figure 1 users utilize their *clients* to post *profiles* or *notifications* (expressed in some appropriate language to be discussed below) to *super-peers*. Clients play a dual role: they can be information producers and information consumers at the same time. The P2P network of super-peers is the "glue" that makes sure that published notifications arrive at interested subscribers. To achieve this, super-peers forward posted profiles to other super-peers using an appropriate P2P protocol. In this way, matching of a profile with a notification can take place at a super-peer that is as close as possible to the origin of the incoming document. Profile forwarding can be done in a sophisticated way to minimize network traffic e.g., no profiles that are less general than one that has already been processed are actually forwarded.

In their capacity as information producers, clients can also post *advertisements* that describe in a "concise" way the notifications that will be produced by them. These advertisements can also be forwarded in the P2P network of super-peers to *block* the forwarding of *irrelevant* profiles towards a source. Advertisement forwarding can also be done in a sophisticated way using ideas similar to the ones for profile forwarding [5].

3. SUMMARY OF RESULTS

Our work on P2P networks for SDI has been driven by the following considerations:

- The next generation of P2P data sharing systems should be developed in a *principled* and *formal* way and classical results from logic and database theory [1] should be applied. Most of the current work on P2P systems has not emphasized such theoretical considerations at all, as witnessed by the recent survey of [10].
- Performance and scalability must be a primary consideration in the design of any realistic system that supports SDI with P2P networks.
- Implementations of P2P systems should be based on modern technologies that would enable rapid application development and reuse.

In the rest of this paper we discuss our work up to now and what we have done to address these considerations.

3.1 Data models and query languages

We have developed the data models $\mathcal{WP}$, $\mathcal{AWP}$ and $\mathcal{AWPS}$, and their corresponding languages for specifying queries and notifications. $\mathcal{WP}$ is based on free text and its query language is based on the *boolean model with proximity operators* as known in the Information Retrieval (IR) community [7]. Data model $\mathcal{AWP}$ is based on *attributes* with values of type text, and its query language is an extension of the query language of data model $\mathcal{WP}$. Finally, the model $\mathcal{AWPS}$ extends $\mathcal{AWP}$ by introducing a "similarity" operator based on the IR vector space model [29].

The following three queries demonstrate the features of $\mathcal{WP}$, $\mathcal{AWP}$ and $\mathcal{AWPS}$ respectively and their use in an SDI application for a digital library:

$$Smith \ \wedge \ (peer\text{-}to\text{-}peer \vee (selective \prec_{[0,0]} dissemination \prec_{[0,3]} information))$$

$$AUTHOR \sqsupseteq Smith \ \wedge \ TITLE \sqsupseteq (peer\text{-}to\text{-}peer \vee (selective \prec_{[0,0]} dissemination \prec_{[0,3]} information))$$

$$AUTHOR \sqsupseteq Smith \wedge ABSTRACT \sim_{0.8} \text{"}Peer\text{-}to\text{-}peer\ architectures\ have\ been...\text{"}$$

Our data modelling work complements recent proposals for querying textual information in SDI systems [5, 4] by using linguistically motivated concepts such as *word* and traditional IR operators (instead of strings and their operators). Our data models and query languages are more expressive than the one used in the centralized SDI system SIFT [29] where documents are free text and queries are conjunctions of keywords. On the other hand, we have only considered notifications that have *flat* structure, thus we cannot support hierarchical documents as in the XML-based models of [2, 6]. But notice that IR-inspired constructs such as proximity and similarity *cannot* be expressed in the query languages of [2, 6] and are also *missing* from W3C standard XML query languages XQuery/XPath. The recent W3C working draft [21] is expected to pave the way for the introduction of such features in XQuery/XPath. We expect our work on models $\mathcal{WP}$, $\mathcal{AWP}$ and $\mathcal{AWPS}$ to serve as a guide for future work on these features in the XML framework. The similarity concept of $\mathcal{AWPS}$ has also been used in database systems with IR influences (e.g., WHIRL [9]) and more recently in the XML query language ELIXIR [8]. We note that both WHIRL and ELIXIR target information retrieval and integration applications, and pay no attention to information dissemination and the concepts/functionality needed in such applications.

3.2 Theoretical complexity analysis

In [20, 19] we study the complexity of the following three algorithmic problems for models $\mathcal{WP}$ and $\mathcal{AWP}$. The first problem is the *satisfiability problem*: deciding whether a profile can ever be satisfied by an incoming notification. The second problem is the *filtering problem*: Given a database of satisfiable profiles db and a notification n, find all profiles $q \in db$ that match n. This functionality is very crucial at each super-peer because we expect deployed information dissemination systems to handle hundreds of thousands or millions of profiles. The third problem is the *entailment problem*: Deciding whether a profile is more or less "general" than another. This functionality is crucial if we want to minimize profile forwarding as sketched in Section 2.

Our results show that the satisfiability and entailment problem for arbitrary queries in $\mathcal{WP}$ and $\mathcal{AWP}$ is NP-complete and coNP-complete respectively. Luckily, these problems can be solved in PTIME for queries in appropriately defined DNF forms. The filtering problem is also solvable in PTIME in both of the above cases [20].

3.3 Efficient Filtering Algorithms

In [19, 27, 26] we study the problem of filtering for conjunctive queries in $\mathcal{AWP}$ and present efficient main-memory profile indexing algorithms. Figures 2 and 3 contrast three of these algorithms: BF (the obvious brute force algorithm), SingleWordIndex (which utilizes a 2-level index: an array

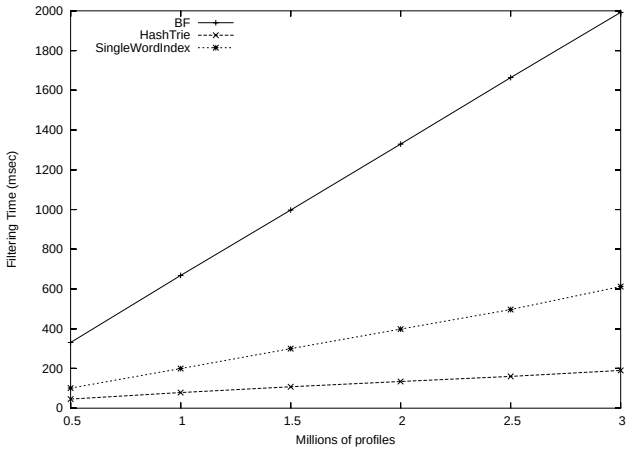

Figure 2: Filtering time for some algorithms for $\mathcal{AWP}$

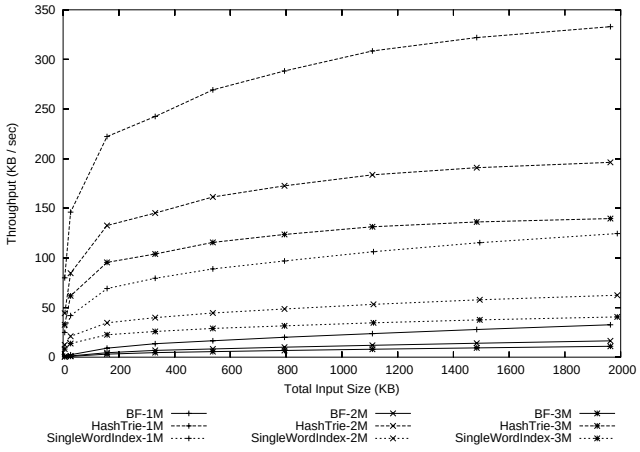

Figure 3: Throughput for some algorithms for $\mathcal{AWP}$

for the attribute names containing pointers to inverted lists for the $\mathcal{WP}$ expression associated with each attribute) and HashTrie (which uses a hash table pointing to *word tries* that are used for clustering similar profiles). HashTrie is a variation of the algorithm Tree of [28]. The three algorithms have been tested in a digital library SDI application using documents downloaded from *ResearchIndex*[1] and profiles consisting of technical terms extracted from the documents. The graph of Figure 2 gives us the time we need to discover matching profiles for a single incoming document while the graph of Figure 3 gives us the throughput achieved by these algorithms. As we can see, HashTrie gives the best filtering performance and, for a given database of 3 million profiles, it can process a stream of around 150 KB of incoming data (around 5 Research Index papers) per second.

4. THE SYSTEM P2P-DIET

In [18] our group has proposed DIAS, a distributed SDI system for digital libraries that follows the architecture of Figure 1 and employs conjunctive queries in $\mathcal{AWPS}$. Work on DIAS has resulted in the implementation of a larger prototype system called P2P-DIET that combines ad-hoc querying as found in other super-peer networks [30] and SDI as proposed in DIAS and discussed above [14].

4.1 Functionality

There are two kinds of nodes in P2P-DIET: *super-peer* nodes and *client* nodes. All super-peers are equal and have the same responsibilities, thus the super-peer subnetwork is a *pure* peer-to-peer network (in terms of graph theory, it can be an arbitrary undirected graph). Each super-peer serves a fraction of the clients and keeps indices on the resources of those clients. A client node can run on the computer of a user. Resources (e.g., files in a file-sharing application) will be kept at client nodes, although it is possible in special cases to store resources at super-peer nodes (see Section 4.3). Clients are equal to each other only in terms of download. When a client wants to actually download a resource, it downloads it directly from the resource owner client. A client node can be connected to the network through a single super-peer node, which is the *access point* of the client. It is not necessary for a client to be connected to the same access point continuously i.e., client migration is allowed. Clients can connect, disconnect or even leave from the system silently at any time. To enhance the general form of the network, we also allow clients to use dynamic IP addresses.

Clients may *publish* a resource by sending a *notification* to their access point. Among other things, a notification contains *metadata* for the resource expressed as conjunctions of attribute-value pairs in the model $\mathcal{AWPS}$. In the file-sharing application which currently runs on top of P2P-DIET, a resource can be a file of any type, for example, a music file or a Microsoft Word document.

P2P-DIET supports the typical *ad-hoc query scenario* of super-peer networks [30]. A client can send a query to its access point and the access point will broadcast this query to all super-peers. In this way, *answers* will be produced for all matching network resources. Answers are returned to the access point of the client originating the query and are then passed to the client for further processing.

P2P-DIET also supports SDI scenarios. Clients may *subscribe* to the system with a *profile* (or *long-standing query*) expressing their information needs. Whenever a *notification* is generated at any point in the network, P2P-DIET makes sure that clients with profiles matching this notification are notified. A high-level view of the P2P-DIET architecture is shown in Figure 4. The concept of advertisements as explained in Section 2 has not yet been implemented.

4.2 Routing

We expect the super-peer subnetwork of P2P-DIET to be more stable than typical pure peer-to-peer networks such as Gnutella, thus we choose to use routing algorithms appropriate for such networks. P2P-DIET implements routing of ad-hoc queries or profiles and notifications by utilizing *minimum weight spanning trees* for the super-peer subnetwork, a *poset* data structure encoding profile entailment as originally suggested in [5], and *very fast profile indexing* at each node using the algorithms of Section 3.3. Preliminary evaluation of these techniques by us (and previously by SIENA

[1] `http://www.researchindex.org`

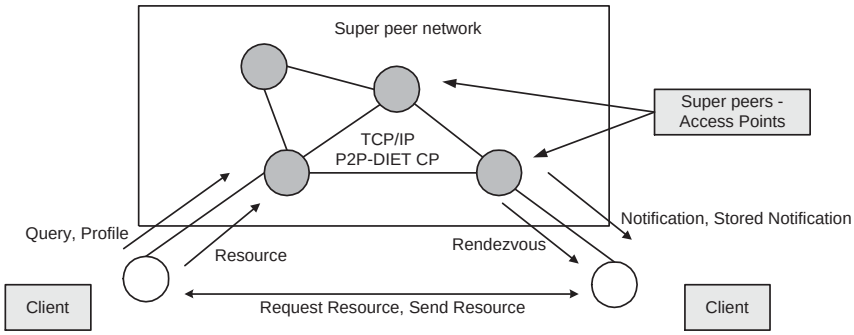

Figure 4: P2P-DIET architecture

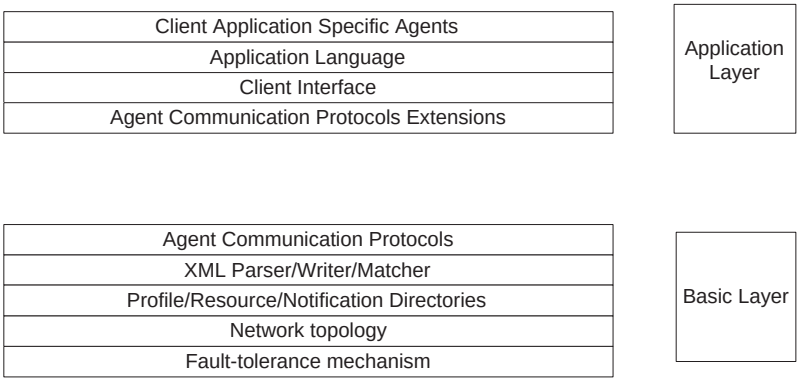

Figure 5: A layered view of P2P-DIET

researchers [5]) show that P2P-DIET is a scalable super-peer system.

4.3 Stored notifications and rendezvous

Clients may not be online all the time, thus we can not guarantee that the client will be available at the time that matching resources are added to the network and relevant notifications are generated. We cannot ignore such situations and allow the loss of relevant notifications.

When a client is off-line, notifications matching its profiles are stored in the *stored notifications directory* of the access point of the client. Stored notifications are delivered to the client the next time that it connects to the network. This happens even if the client has migrated to a different access point.

A similar situation is when a client A requests a resource, but the resource owner client B is not online. In that case, the client A may request a *rendezvous* with the resource. When client B later-on reconnects to the network, its access point informs it that the resource must be delivered to the access point of the client A as a *rendezvous resource*. It is stored in the *rendezvous directory* of this super-peer and the next time that the client A connects to the network, it receives a rendezvous notification. This takes place even if client A has migrated to a different access point. Client A may request the resource directly from its current access point or its previous one in case of migration.

4.4 Supporting network topology changes

Clients may use dynamic IP addresses. We need a way to identify clients and we cannot use their IP address to do this because the address of a client may be different each time it connects to the Internet. Client identification in P2P-DIET is done using unique keys assigned by the super-peer that the client connects to for the very first time. When a new client wants to register to the network, it connects to a super-peer and this super-peer assigns a unique key to the client.

Every time a client wants to connect to the network after registration, it must use its key. The client sends a `connect` message along with its key to its access point. The super-peer then updates the client entry with that key in the client list. When a user disconnects or equivalently terminates the client application program, its access point must be aware of that. The client sends a `disconnect` message along with its key to its access point and the super-peer updates the client entry with that key in the client list.

P2P-DIET supports client migration to different super-peers without loosing useful information like stored notifications or resource rendezvous as we explained in Section 4.3. In a

future version of P2P-DIET, where a load balancing mechanism will be included, the client migration process might prove useful for moving clients from overloaded to underloaded super-peers.

Another important need in an Internet scale peer-to-peer network, is the ability to add new super-peers (e.g., because of an ever increasing number of clients). When a new super-peer joins the super-peer network, the minimum weight spanning trees for each node need to be updated. In addition, the new super-peer needs to be informed about profiles of neighboring super-peers that have been previously forwarded.

4.5 Fault-tolerance

Super-peers may fail in any way. For example, a crash on the system or even running out of resources (overloaded - cannot handle any more sockets), are two basic reasons for a super-peer to fail or freeze temporarily. The proper way to solve that problem, is to make every super-peer check periodically whether its neighbor super-peers are alive and are working properly using `are-you-alive` messages. When a super-peer realizes that one of its neighbors is not responding, it broadcasts a `build-spanning-tree` message and the network automatically enters a re-organization phase until all routing paths have been updated.

Clients may face system failures too or may leave from the network silently. A super-peer periodically checks, if all the clients that it serves are alive. If a client is not responding to `are-you-alive` messages the super-peer will treat the situation as if the client had sent a `disconnect` message.

4.6 Implementation

P2P-DIET has been developed as a *general* query and notification service on top of the mobile agent system DIET Core presented in [13]. For a general discussion of what agent systems research has to offer to P2P computing and vice versa the interested reader is invited to see [15, 22].

A layered view of P2P-DIET is shown in Figure 5. A file-sharing application has been built on top of P2P-DIET to demonstrate its features.

5. CONCLUSIONS

We have argued that it is natural and useful to add SDI functionality to current super-peer P2P systems analyzed in [30]. We have summarized our work on SDI, and discussed the details of our prototype system P2P-DIET that integrates ad-hoc querying and SDI functionality in a single super-peer P2P system. Currently we are working on implementing the query and SDI functionality of P2P-DIET on top of a distributed hash table [24] and compare this with our current implementation.

In collaboration with Peter Triantafillou's group at the University of Patras we have also recently studied the problem of load balancing in a hierarchical P2P system as a *fair resource allocation* problem [25]. In related work, we currently study distributed algorithms for load balancing in *task-sharing* P2P systems using ideas from physics (diffusion) and nature-inspired computing (ant colonies). These ideas are very natural if one considers P2P systems to be *complex adaptive systems* [3, 16].

Acknowledgements

We would like to acknowledge the comments of all members of project DIET on various aspects of our work. Finally, we would like to thank Evangelos Milios of Dalhousie University for providing us with the document collection we used for testing our profile indexing algorithms.

6. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[2] M. Altinel and M. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.

[3] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A framework for the development of agent-based peer-to-peer systems. In *Proceedings of IEEE International Conference on Distributed Computer Systems*, pages 15–22, 2002.

[4] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficent filtering in publish-subscribe systems using binary decision diagrams. In *Proc. of 23rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001.

[5] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[6] C.-Y. Chan, P. Felber, M. Garofalakis, and R. Rastogi. Efficient Filtering of XML Documents with XPath Expressions. In *Proceedings of the 18th International Conference on Data Engineering*, pages 235–244, February 2002.

[7] C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. *ACM Transactions on Information Systems*, 17(1):1–39, 1999.

[8] T. T. Chinenyanga and N. Kushmerick. Expressive retrieval from XML documents. In *Proceedings of SIGIR'01*, September 2001.

[9] W. W. Cohen. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118(1-2):163–196, 2000.

[10] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data sharing peer-to-peer systems. In *Proceedings of the 9th International Conference on Database Theory (ICDT 2003)*, volume 2572 of *Lecture Notes in Computer Science*, pages 1–15. Springer, January 2003.

[11] M. J. Franklin and S. B. Zdonik. "Data In Your Face": Push Technology in Perspective. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 516–519, 1998.

[12] B. Gedik and L. Liu. PeerCQ: A Decentralized and Self-Configuring Peer-to-Peer Information Monitoring System. In *Proceedings of the 23rd IEEE International Conference on Distributed Computer Systems*, May 2003.

[13] C. Hoile, F. Wang, E. Bonsma, and P. Marrow. Core specification and experiments in DIET: a decentralised ecosystem-inspired mobile agent system. In *Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2002)*, pages 623–630, July 15–19 2002.

[14] S. Idreos and M. Koubarakis. P2P-DIET: A Query and Notification Service Based on Mobile Agents for Rapid Implementation of P2P Applications. Technical report, Intelligent Systems Laboratory, Dept. of Electronic and Computer Engineering, Technical University of Crete, 2003.

[15] M. Koubarakis. Multi-agent Systems and Peer-to-Peer Computing: Methods, Systems and Challenges. Invited talk at the 7th International Workshop on Cooperative Information Agents (CIA 2003), Helsinki, Finland, 27-29 August 2003. Paper available at `http://www.intelligence.tuc.gr/~manolis`.

[16] M. Koubarakis. Complex Adaptive Systems Research for Peer-to-Peer Computing. Manuscript in Preparation.

[17] M. Koubarakis. Boolean Queries with Proximity Operators for Information Dissemination. Proceedings of the Workshop on Foundations of Models and Languages for Information Integration (FMII-2001), Viterbo, Italy , 16-18 September, 2001. In LNCS (forthcoming).

[18] M. Koubarakis, T. Koutris, P. Raftopoulou, and C. Tryfonopoulos. Information Alert in Distributed Digital Libraries: The Models, Languages and Architecture of DIAS. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2002)*, volume 2458 of *Lecture Notes in Computer Science*, pages 527–542, September 2002.

[19] M. Koubarakis and C. Tryfonopoulos. Peer-to-peer agent systems for textual information dissemination: algorithms and complexity, December 2002. In the *UK Workshop on Multiagent Systems (UKMAS-2002)*, Liverpool, UK.

[20] M. Koubarakis, C. Tryfonopoulos, P. Raftopoulou, and T. Koutris. Data models and languages for agent-based textual information dissemination. In *Proceedings of 6th International Workshop on Cooperative Information Systems (CIA 2002)*, volume 2446 of *Lecture Notes in Computer Science*, pages 179–193, September 2002.

[21] XQuery and XPath Full-Text Use Cases. W3C Working Draft 14 February 2003. Available at `http://www.w3.org/TR/xmlquery-full-text-use-cases`.

[22] G. Moro and M. Koubarakis, editors. *Agents and Peer-to-Peer Computing (Proceedings of First International Workshop, AP2PC 2002)*, volume 2530 of *Lecture Notes in Computer Science*. Springer, 2003.

[23] P. R. Pietzuch and J. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. Proceedings of the International Workshop on Distributed Event-Based systems (DEBS'02), July 2-3, 2002, Vienna, Austria.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[25] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high-performance peer-to-peer content and resource sharing systems. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, January 2003.

[26] C. Tryfonopoulos, Y. Drougas, and M. Koubarakis. Efficient and Scalable Selective Information Dissemination using Data Models Inspired from Information Retrieval. Manuscript in Preparation.

[27] C. Tryfonopoulos and M. Koubarakis. Agent-based textual information dissemination: Data models, query languages, algorithms and computational complexity. Technical report, Intelligent Systems Laboratory, Dept. of Electronic and Computer Engineering, Technical University of Crete, 2002.

[28] T. Yan and H. Garcia-Molina. Index structures for selective dissemination of information under the boolean model. *ACM Transactions on Database Systems*, 19(2):332–364, 1994.

[29] T. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, 1999.

[30] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, March 5–8 2003.