Data Sharing and Querying for Peer-to-Peer Data Management Systems

Anastasios Kementsietsidis

Department of Computer Science,
University of Toronto, Canada
tasos@cs.toronto.edu

Abstract. In this work, we investigate mechanisms to support data sharing and querying in a *peer-to-peer data management system*, that is, a peer-to-peer system where each peer manages its own data. To support data sharing, we propose the use of mapping tables which list pairs of corresponding data values that reside in different peers. Our work illustrates how automated tools can help manage the tables between multiple peers by inferring new tables from existing ones and by checking their consistency. In terms of querying, we propose a framework in which users pose queries only with respect to their local peer. Then, we provide a rewriting mechanism that uses mapping tables to translate a locally expressed query to a set of queries over the acquainted peers.

1 Introduction

Peer-to-peer computing consists of an open-ended network of distributed computational peers, where each peer exchanges data and services with a set of other peers, called its acquaintances. The peer-to-peer paradigm, initially popularized by file-sharing systems such as Napster [1] and Gnutella [2], offers an alternative to traditional architectures found in distributed systems and the web. Distributed systems are rich in services but require considerable overhead to launch and have a relatively static, controlled architecture. In contrast, the web offers a dynamic, anyone-to-anyone architecture with minimum startup costs but limited services. Combining the advantages of both architectures, peer-to-peer offers an evolving architecture where peers come and go, choose with whom they interact, and enjoy some traditional distributed services with less startup cost.

Each peer-to-peer system provides two basic services to its peers. First, it offers to its peers the ability to share data with each other. Second, it offers the ability for peers to query each other's contents. Our objective is to investigate mechanisms that can be used to support these two services in the context of a *peer-to-peer data management system*, that is, a peer-to-peer system where each peer manages its own data. In the past, database research has dealt with similar issues in the context of multidatabase systems [10]. However, the solutions provided there are not directly applicable to peer-to-peer data management systems. This is mainly due to the following three features of the new paradigm: the lack of centralized control; the transience of the inter-peer connections; the limited cooperation among the peers.

Traditionally, in multidatabase systems, data integration and exchange between heterogeneous data sources is provided mainly through the use of inter-schema mappings (generalized view definitions often called global-or-local-as-view (GLAV) mappings) that specify how the schemas of the sources are related [17, 21]. Given two sources S_1 and S_2, to construct a mapping m between them, the sources must be willing to share at least portions of their schemas and cooperate in establishing and managing the mappings. Management of mappings becomes an issue when the participating sources change their schemas [24]. Then, the mappings must be updated to reflect the changes. Our work considers peer-to-peer settings in which cooperation between sources to establish such schema mappings is either not desirable (perhaps for privacy reasons) or not feasible (since sources might belong to different worlds [18]). In such settings, we consider an alternative and more flexible form of mappings, called *mapping tables*. Mapping tables are data-level mappings which list pairs of corresponding values between two sources. Our work motivates the use of mapping tables in peer-to-peer environments and shows that the maintenance of these tables can be automated [15]. In terms of query answering, existing approaches assume that a query posed on source S_1 can be answered locally (using S_1's database) and in addition, it can be translated, using the schema-level mapping m, into a query on S_2. However, this environment makes the (implicit) assumption that the answer to the query that S_2 provides can be made to conform to the schema of S_1. Our work makes no such assumption but relies on mapping tables to translate structured queries between peer databases that may contain related data that overlaps little, if at all [14].

Contributions

To the best of our knowledge, our work is the first to address the problem of *data sharing* between heterogeneous sources. Data sharing deals with the exchange of data between heterogeneous sources whose data need not be interdependent and may represent different real world domains. It may not be possible or desirable to transform such data to fit a common schema. As such, data sharing differs from the well-studied problems of data integration [17] and data exchange [11]. Both of the latter two problems use schema-level mappings to express the relationships between source schemas. In data integration, these mappings are used, at run time, to conform the data of one source to the schema of another. In data exchange, the mappings are used to populate a target schema with the data of a source schema.

Our work investigates the use of mapping tables as the primary mechanism for data sharing between peers [15]. Currently, mapping tables provide the basis for data sharing in peer-to-peer data management systems such as the ones found in the domain of biological databases [9]. Still, the creation of mapping tables is a time-consuming and manual process performed by a set of expert curators. We are aware of no data management tools currently designed to facilitate the creation, maintenance and management of these tables. Our work illustrates how automated tools can help manage mappings between multiple sources by inferring new mappings and checking their consistency. In practice, inferring new mappings proves useful when two peers first become acquainted. The system is able to use existing mappings in the network of peers to associate the values of the newly acquainted peers. This task requires no human intervention and, thus, it alleviates the need for a human curator. Checking consistency of mappings is important in order to detect and report possible errors in existing mappings. Without automated support, curators edit, copy, or merge mappings that come from a variety of sources and it can be a cumbersome task to make sure that the associations of one mapping do not *conflict* with the ones expressed by another. Our approach offers an automated way to detect these inconsistencies.

In terms of query answering, we propose a framework in which users pose queries only with respect to their local peer schema. Then, we provide a rewriting mechanism that uses mapping tables to translate a locally expressed query to a set of queries that can be executed in the acquainted peers. Although the idea of query translation is not novel, the context in which it is applied is. Traditional views or schema mappings are queries (or pairs of queries), so query translation involves manipulating relatively small queries. Mapping tables however contain data and may be very large. Query translation in our environment involves manipulating these large tables.

An important contribution of our work is the use of a common formalism for both data sharing and for query answering. Through this formalism, we are able to represent both mapping tables and queries. This uniformity offers a number of advantages including the ability to store in a peer database both queries and their translations and the ability to reuse algorithms that were developed for our data sharing setting during query answering.

For the purposes of our work, and in order to test the applicability of our ideas, we have implemented our solutions on a prototype peer-to-peer data management system in which each peer is a data management system with its own schema and data. Peers communicate using a Gnutella-like protocol which is customized to our specific needs. For our experiments, we use two different domains, namely, biological databases and flight reservation information.

Outline

Section 2 offers an overview of the database literature on peer-to-peer systems. Section 3 presents an overview of our main results and the paper concludes in Section 4 with a discussion of our future work. The work presented in this article is part of the Hyperion project[1] [6] at the University of Toronto.

2 State of the Art

To put our work in perspective, it proves useful to introduce a classification of existing peer-to-peer systems. We classify such systems into two categories based on the behaviors and characteristics of the peers within each system. Distinguishing characteristic of the peers in the first system category is that these are *altruistic*. In more detail, systems in this category often rely on the architecture and the services of systems like CAN [22], CHORD [23] or Tapestry [25]. An example of such a system is OceanStore [3] while relies on Tapestry to provide distributed file storage. A peer that joins such systems offers its computational or storage resources to it. The system often decides the set of peers with which the new peer will be acquainted. Furthermore, the system can decide the data

[1] In Greek mythology, Hyperion [high-peer-ee-on] is the Titan of light. His name literally means "dweller on high" or "the one above". He was the father of Helius (the sun), Selene (the moon), and Eos (the dawn).

contents of the peer and where these data should be replicated (to increase availability). Due to this replication, the data stored in a peer are available even after the particular peer leaves the system. In term of research, some of the main issues here are the efficiency of the lookup and routing services, and how to maintain this efficiency in the presence of peer arrivals and departures.

In the second category of systems, peers are *selfish*. Here, each peer brings into the system its own data and decides independently with which of the other peers it will be acquainted. Once the peer leaves the systems, its data usually become unavailable to the other peers. Our own work in the context of the Hyperion project, is classified in this second category of systems. Our main focus is on the *data management* issues that arise in this environment. As an example, we are interested on the issue of heterogeneity. While in the previous category of systems, peers are assumed to have the same underlying format (or schema) to represent their data, here each peer might use a different data representation. We do not consider the efficiency of routing schemes and the scalability of our solutions to tens of thousands of peers. One reason for this is that the size, in terms of number of peers, of the systems we consider is considerably smaller.

In what follows we review some of the work in peer-to-peer data management which is the main focus of this thesis. Bernstein et al [7] introduce the Local Relational Model (LRM), a data model designed for peer-to-peer applications. The model's aim is to support semantic interoperability between relational databases in the absence of a global schema. The proposed model makes use of *domain relations* which are equivalent to the notion of mapping tables. However, no provision is described to manage these relations.

Lenzerini [17] describes a general framework for modeling data integration applications which can also be used to represent peer-to-peer applications. An important difference between this work and ours is that the former uses schema-level mappings between the peers, while our work relies mainly of data-level mappings. Thus, the two works are complimentary.

Our initial work on mapping tables shows how these can be used in support of keyword-based searches within peer-to-peer systems. Our current work extends these results to consider the use of mapping tables to support structured queries. The work of Huebsch et al [13] also supports structured queries but the assumption there is that all peers share the same schema.

In Piazza, associations between peers are expressed through either global-as-view (GAV) or local-as-view (LAV) mappings [12, 19]. During query answering, both types of mappings are used to translate queries between peers. Our work on query translation is complimentary to Piazza since our solutions do not rely on GAV/LAV mappings but assume that the only associations available between peers are in the form of mapping tables. An important distinction between the two proposals is that in Piazza the data retrieved from the peer-to-peer network always conform to the schema of the peer where the user-query is initiated. Our work, on the other hand, makes no such assumption.

Ng et al [20] propose an alternative approach to query translation in peer-to-peer environments. Initially, the authors assign a set of descriptive keywords to each schema element of a peer schema. Then, the schema elements of different peers are associated if they have a *similar* set of descriptive keywords. Once schema element associations are established, the translation of queries between different peers is performed by using associated schema elements. An important limitation of the approach is the underlying assumption that descriptive keywords are used consistently throughout the peer-to-peer network. Thus, unlike our work, their solutions cannot handle differences in the vocabularies of the peers.

The work of Chang and Garcia-Molina [8] deals with the translation of queries between heterogeneous sources. The authors use syntactic rules to map selection predicates from one database to that of another. At first glance, mapping tables look like *materializations* of these syntactic rules. However, the two constructs operate under different assumptions. As an example, a syntactic rule that maps two selection predicates ignores the intricacies of this mapping at the data level, that is, the fact that at that level the mapping between values might be incomplete or be many-to-many. Part of our work on mapping tables addresses exactly these issues. Our approach also offers a *uniform* representation both for the rules, i.e., the mappings at the data level, the queries, and for the mappings between translated queries.

Aberer et al [4] introduce a formal framework to assess the quality of peer mappings by measuring the quality of query rewritings that are obtained from these mappings. The mappings considered by the authors are similar, in spirit, to mapping tables since they have the form of functions that map the values of attributes belonging to acquainted peers. However, the focus of this work is more on the assessment of the quality of the mappings, and less on the mappings themselves and how these can be used to perform rewriting. Our work, on the other hand, focuses on these latter issues. Thus, the two approaches are complimentary.

3 Status of Current Work

This work was developed within the framework of the Hyperion project [6] at the University of Toronto. Thus, before we move on to the main topics of the thesis, we offer an overview of Hyperion and we present its main design principles. This, in turn, allows us to put our solutions in context since it shows the environment within which our solutions are designed to work.

3.1 The Hyperion Architecture

The objective of the Hyperion project is to facilitate data sharing between autonomous and heterogeneous sources that are organized in a peer-to-peer fashion. Source autonomy is an important requirement in practice. Sources that join a peer-to-peer network are usually not willing to compromise their autonomy while doing so. As an example, consider biological data sources where each peer source belongs to a different research group or institute. Although each group is willing, due to mutual benefit, to share its data with other groups, it is often not willing to alter its data representation or its internal naming scheme. Which brings us to the next issue, namely, heterogeneity, as a by-product of source autonomy.

In such settings, the approach taken by Hyperion to address the above issues is to augment each source with an interoperability layer. The layer, shown in Figure 1, consists mainly of three modules which we briefly describe here:

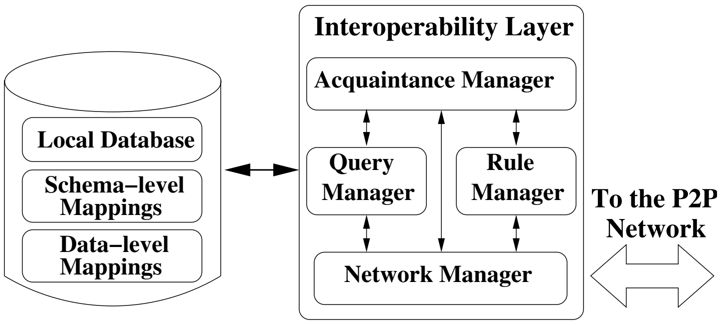

Fig. 1. The Hyperion interoperability layer

- **Acquaintance Manager:** The objective of the acquaintance manager is to resolve heterogeneity issues that arise between a pair of peer sources that are acquainted. To this end, the module maintains any possible schema-level (GLAV) mappings between the schemas of the sources. However, since such mappings are not always available, or possible, it also maintains possible data-level mappings, in the form of mapping tables. Mapping tables respect peer autonomy since they do not impose constraints on the peer schemas. They are minimally invasive in that they do not restrict the operation of peers in anyway beyond the agreement on values expressed in the tables. The first part of our thesis deals with the maintenance of these tables within the acquaintance manager.

- **Query Manager:** One of the guiding principles in Hyperion is that a user should only be aware and knowledgeable of her own local peer schema. As a result, all user queries are expressed with respect to this schema and it is the responsibility of the system to retrieve additional related results from the peer-to-peer network. The query manager is responsible for rewriting the locally expressed queries to ones that can be executed with respect to the schemas of other peers. The second part of our thesis focuses on how this translation can be performed through the use of mapping tables.

- **Rule Manager:** The objective of the rule manager is to provide mechanisms that can be used by peers in order to coordinate updates between them. Our proposal is to achieve such coordination through the use of Event-Condition-Action (ECA) rules. Providing update coordination in a peer-to-peer environment presents some interesting technical challenges. However, since this work in not part of our thesis, we will not elaborate any further.

In what follows, we provide more details about our solutions for the first two modules of Hyperion.

3.2 Mapping Tables

Mapping tables are the primary mechanism for data sharing between peers. In what follows we offer an overview of our work on mapping tables [15, 16].

Consider two peers S_1 and S_2 that expose attributes U and V respectively. Then, a mapping table m is a relation over the attributes $X \cup Y$, where $X \subseteq U$ and $Y \subseteq V$ are non-empty sets of attributes from the two peers. A tuple (x, y) in the mapping table indicates that the value x, in the domain of X, is associated with the value y, in the domain of Y. We impose no restrictions on the association between values, and thus the recorded associations can be one-to-one, one-to-many or many-to-many.

Starting from this simple idea of associating values between different domains, our framework offers a number of extensions including the ability to use variables. Variables offer a compact and convenient way to represent common associations between values, such as, the identity association. For example, a tuple (v_1, v_1) in mapping table m, where v_1 denotes a variable, represents the fact that every value in the domain of X is associated to itself in the domain of Y. The introduction of variables in mapping tables necessitates the use of valuations [5] in order to determine the set of Y-values with which a certain X-value is associated. Formally, given a valuation ρ of the variables of mapping table m, a value $x \in \pi_X(\rho(m))$ is associated with a certain set of values in the domain of Y, namely, with the set $\pi_Y(\sigma_{X=x}(\rho(m)))$.

Up to this point, we concerned ourselves only with the X-values appearing in the mapping table and how these are associated. What about the X-values that do not appear in the table? What can we say about their associations? Our work investigates alternative semantics for mapping tables to address these and related issues. Through the alternative semantics, we are able to specify whether the missing X-values can be associated either to any Y-value (open-world semantics) or to no Y-value (closed-world semantics).

3.3 Mapping Constraints

Mapping tables, by definition, represent value correspondences. Part of our contribution is the treatment of mapping tables as constraints on the exchange of information between sources. This is achieved by using the associations of values, recorded in the tables, to constrain the association of tuples in the two peers. Formally, consider again peers S_1 and S_2 and let $t_1[U]$ and $t_2[V]$ denote tuples of the two peers, respectively. Then, tuple t_1 with $t_1[X] = x$ can be mapped, with respect to mapping table m and valuation ρ, only to tuples t_2 in peer S_2 for which $t_2[Y] \in \pi_Y(\sigma_{X=x}(\rho(m)))$.

An important benefit of *mapping constraints* is our ability to automatically maintain and combine multiple constraints over a network of peers. Still, an obstacle that hinders the immediate deployment of our techniques is the high complexity of the inference and consistency problems for mapping constraints. Specifically, our investigation shows that the inference problem for mapping constraints is NP-complete even under restrictions that can be considered severe. To reduce the complexity of the problem, we provide a solution that applies over paths. We perform inference over these paths and we show experimentally that our algorithm works efficiently in practice. Furthermore, our experiments show that there is added benefit to consider alternative paths in a peer-to-peer network since we are able, in practice, to infer additional data associations.

3.4 Query Translation

Our work on peer-to-peer query answering starts with the investigation of the query answering semantics [14]. In brief, the proposed semantics allow for query answers that need not conform to a common schema. As such, our query semantics are consistent with the requirements of the data-sharing problem.

The execution of queries over the peer-to-peer network requires the translation of queries between peers. This translation relies solely on the use of mapping tables. Specifically, in our work we show how mapping tables can be used to translate select-project-join queries, where the selection formula is *positive*, i.e., it has no negation and it consists of conjunctions and disjunctions of atoms of the form $(A = B)$ and $(A = a)$, where A and B are attribute names and a is a constant. We consider both sound translations (which only retrieve correct answers) and complete translations (which retrieve all correct answers, and no incorrect answers). In this setting, we investigate the complexity of testing for sound translations and we show that the problem is Π_2^p-complete, in the size of the query. Since large queries rarely occur in practice, the high complexity is not an obstacle and as evidence we provide an implementation of the algorithm that works efficiently. We also propose and implement algorithms for computing sound and complete translations, and we offer experimental results that show the efficiency of these algorithms.

One of the advantages of our approach is that we use the same underlying formalism to represent both mapping tables and queries. Specifically, we introduce the notion of T-queries which is a tabular representation of queries and we show that for each select-join query, where the selection formula is positive, we can have an equivalent T-query. Our solutions deal with projection independently since we show that the issues involved in projection are orthogonal to those for the other two operators. The introduction of T-queries offers a number of advantages including simplicity of implementation of the proposed algorithms due to the uniformity of representation. Furthermore, by representing queries as tables, we are able to store as part of our database both the queries themselves, and the query translations. Our system attempts to take advantage of these past, stored, translations during the process of translating a new query. The objective is to reuse, if possible, the stored translations to reduce the time to compute a translation for the query under consideration. Our experiments prove that this technique is effective and results in considerable savings in terms of computation time.

4 Conclusions and Future Work

The objective of this thesis was to investigate mechanisms to support data sharing and querying in peer-to-peer data management systems. To this end, we proposed the use mapping tables as a mechanism to address the heterogeneity of the peers while respecting peer autonomy. We described briefly how automated tools can help manage the tables between multiple peers by inferring new tables from existing ones and by checking their consistency. In terms of querying, we proposed a framework in which users pose queries only with respect to their local peer. Then, we outlined our rewriting mechanism which uses mapping tables to translate a locally expressed query to a set of queries over the acquainted peers.

In terms of future work, there are a number of issues to address in the area of data sharing through the use of mapping tables. Specifically, our current semantics of mapping tables do not accommodate for NULL values and we intend to investigate how NULLs can be incorporated into these semantics.

In the current implementation, the mapping table inference algorithm accepts as input a path of peers and, by using the mapping tables along the path, it computes as output a set of inferred mapping tables. Since the peers on the path are autonomous, they can update their corresponding mapping tables independently and without notifying any other peer. These updates may influence the mapping tables that can be inferred along the path. Currently, to reflect the effect of the updates on the inferred mapping tables, one must re-execute the inference algorithm. Clearly, this approach is not optimal since a large portion of the previously inferred mapping tables remains unaffected by the updates. Thus, what is required is an incremental inference algorithm that detects the updates in the mapping tables of the path and determines which inferred mapping tables are affected by these updates and how these tables need to be updated.

Our query language currently does not support negation. As part of our future work, we intend to investigate how we can incorporate this operator. Such an extension will require a corresponding extension in the semantics of mapping tables. This is because our solutions rely on the same underlying formalism to represent mapping tables and queries.

References

1. Napster. `http://www.napster.com/`.
2. Gnutella. `http://www.gnutelliums.com/`.
3. OceanStore. `http://oceanstore.cs.berkeley.edu/`.
4. Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. The chatty Web: emergent semantics through gossiping. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 197–206. ACM Press, 2003.
5. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
6. Marcelo Arenas, Vasiliki Kantere, Anastasios Kementsietsidis, Iluju Kiringa, Renée J. Miller, and John Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination. *SIGMOD Record*, 32(3):53–58, 2003.
7. Philip Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu. Data Management for Peer-to-Peer Computing: A Vision. In *Proc. of the Int'l Workshop on the Web and Databases (WebDB)*, 2002.
8. Chen-Chuan K. Chang and Hector Garcia-Molina. Mind your vocabulary: Query mapping across heterogeneous information sources. In *ACM SIGMOD Int'l Conf. on the Management of Data*, pages 335–346, 1999.
9. Susan Davidson, G. Christian Overton, and Peter Buneman. Challenges in integrating biological data sources. *Journal of Computational Biology*, 2(4):557–572, 1995.
10. A. Elmagarmid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann Publishers, 1999.
11. Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proc. of the Int'l Conf. on Database Theory (ICDT)*, pages 207–224, 2003.
12. Alon Halevy, Zack Ives, Dan Suciu, and Igor Tatarinov. Schema Mediation in Peer Data Management Systems. In *Proc. of the Int'l Conference on Data Engineering*, 2003.

13. Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham Boon, Thau Loo, Scott Shenker, and Ion Stoica. Querying the Internet with PIER. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 321–332, 2003.
14. Anastasios Kementsietsidis and Marcelo Arenas. Data sharing through query translation in autonomous sources. *(To appear in VLDB 2004)*.
15. Anastasios Kementsietsidis, Marcelo Arenas, and Renée J. Miller. Data mapping in peer-to-peer systems: Semantics and algorithmic issues. In *ACM SIGMOD Int'l Conf. on the Management of Data*, pages 325–336, 2003.
16. Anastasios Kementsietsidis, Marcelo Arenas, and Renée J. Miller. Managing data mappings in the Hyperion project. In *Proc. of the Int'l Conference on Data Engineering*, pages 732–734, 2003.
17. Maurizio Lenzerini. Data Integration: A Theoretical Perspective. In *Proc. of the ACM Symp. on Principles of Database Systems (PODS)*, pages 233–246, 2002.
18. Bertram Ludäscher, Amarnath Gupta, and Maryann E. Martone. Model-based mediation with domain maps. In *Proc. of the Int'l Conference on Data Engineering*, pages 81–90, 2001.
19. Jayant Madhavan and Alon Y. Halevy. Composing Mappings Among Data Sources. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 572–583, 2003.
20. Wee Siong Ng, Beng Chin Ooi, Kian Lee Tan, and Ao Ying Zhou. PeerDB: A P2P-based system for distributed data sharing. In *Proc. of the Int'l Conference on Data Engineering*, pages 633–644, 2003.
21. Lucian Popa, Yannis Velegrakis, Renée J. Miller, Mauricio A. Hernandez, and Ronald Fagin. Translating web data. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 598–609, 2002.
22. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *ACM SIGCOMM Int'l Conf. on Data Communications*, pages 161–172, 2001.
23. Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for Internet applications. In *ACM SIGCOMM Int'l Conf. on Data Communications*, pages 149–160, 2001.
24. Yannis Velegrakis, Renée J. Miller, and Lucian Popa. Mapping adaptation under evolving schemas. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 584–595, 2003.
25. Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.