Karl Aberer
Manolis Koubarakis
Vana Kalogeraki (Eds.)

Databases, Information Systems, and Peer-to-Peer Computing

LNCS 2944

First International Workshop, DBISP2P 2003
Berlin, Germany, September 2003
Revised Papers

Springer

Karl Aberer Manolis Koubarakis
Vana Kalogeraki (Eds.)

Databases, Information Systems, and Peer-to-Peer Computing

First International Workshop, DBISP2P 2003
Berlin, Germany, September 7 – 8, 2003
Revised Papers

Springer

Volume Editors

Karl Aberer
Swiss Federal Institute of Technology (EPFL)
Distributed Information Systems Laboratory
School of Computer and Communication Sciences
1015 Lausanne, Switzerland
E-mail: karl.aberer@epfl.ch

Manolis Koubarakis
Technical University of Crete
Dept. of Electronic and Computer Engineering
Chania 73100 Crete, Greece
E-mail: manolis@intelligence.tuc.gr

Vana Kalogeraki
University of California, Riverside
Dept. of Computer Science and Engineering
Riverside, CA 92521, USA
E-mail: vana@cs.ucr.edu

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): H.2, H.3, H.4, C.2, I.2.11, D.2.12, D.4.3, E.1

ISSN 0302-9743
ISBN 3-540-20968-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10982693 06/3142 5 4 3 2 1 0

Preface

Peer-to-peer (P2P) computing is currently attracting enormous media attention, spurred by the popularity of file sharing systems such as Napster, Gnutella and Morpheus. In 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.

The wealth of business opportunities promised by P2P networks has generated much industrial interest recently, and has resulted in the creation of various industrial projects, startup companies, and special interest groups. Researchers from distributed computing, networks, agents and databases have also become excited about the P2P vision, and papers tackling open problems in this area have started appearing in high-quality conferences and workshops.

Much of the recent research on P2P systems seems to be carried out by research groups with a primary interest in distributed computation and networks. This workshop concentrated on the impact that current database research can have on P2P computing and vice versa. Although researchers in distributed data structures and databases have been working on related issues for a long time, the developed techniques are simply not adequate for the new paradigm. P2P computing introduces the paradigm of decentralization going hand in hand with an increasing self-organization of highly autonomous peers, thus departing from the classical client-server computing paradigm. This new paradigm bears the potential to realize computing systems that scale to very large numbers of participating nodes. Taking advantage of this potential for the area of data management is a challenge that the database community itself is asked to face. The realization of the P2P computing vision is however a Herculean task, fraught with immense technical difficulties. As a result, it offers database theoreticians and system developers a new set of exciting open problems.

We believe that database research has much to contribute to the P2P grand challenge through its wealth of techniques for sophisticated semantics-based data models, clever indexing algorithms and efficient data placement, query processing techniques and transaction processing. The database community could benefit from the P2P computing vision by developing loosely coupled federations of databases where databases can join and leave the network at will; a single global schema is not a possibility, and answers need not be complete but should be best effort.

Database technologies in the new information age will form the crucial components of the first generation of complex adaptive information systems. These are an emerging kind of information systems that are very dynamic, self-organize continously and adapt to new circumstances, they are locally but not globally optimized, and form a whole which is greater than the sum of its parts. These new information systems support highly dynamic, ever-changing, autonomous social organizations and can no longer be developed using traditional analy-

sis, design and implementation techniques. This workshop also concentrated on complex adaptive information systems, their impact on current database technologies and their relation to emerging industrial technologies such as IBM's autonomic computing initiative.

This workshop, collocated with VLDB, the major international database and information systems conference, brought together key researchers from all over the world working on databases and P2P computing with the intention of strengthening this connection. Also researchers from other related areas such as distributed systems, networks, multiagent systems and complex systems participated.

The workshop was jointly organized with the AP2PC workshop which is part of the AAMAS conference and is under the responsibility of the same steering committee. Together these two workshops address both the agent and the database communities and thus take account of the interdisciplinary nature of P2P computing.

The DBISP2P workshop received 32 submissions that entered the review process. All submissions underwent a rigorous review that was completed by an online PC discussion for making the final selection of 16 papers. The organizers would like to thank at this point all program committee members for their excellent work. The program was completed by a keynote speech and a panel. The keynote speech with the title "Design Issues and Challenges for RDF- and Schema-Based Peer-to-Peer Systems" was presented by Wolfgang Nejdl from the University of Hannover. Aris Ouksel organized a panel on the topic "P2P Computing and Database Technologies: Convergence of Technologies and Socio-economic Characteristics on the Web, Benefits and Technical Challenges in Database Applications" with the goal to explore the promise of P2P to offer exciting new possibilities in distributed information processing.

The organizers would particularly like to thank Klemens Böhm from the University of Magdeburg for his excellent work in taking care of the local arrangements, the VLDB organization for their valuable support of the workshop organization, and the steering committee for the opportunity to set up this workshop and for their continuing support.

September 2003 Karl Aberer, Manolis Koubarakis, Vana Kalogeraki

Organization

The 1st International Workshop on Databases, Information Systems and Peer-to-Peer Computing took place at Humboldt University, Berlin, Germany on September 7–8, 2003, collocated with VLDB 2003.

Workshop Chairs

Program Chairs	Karl Aberer (EPFL, Lausanne, Switzerland) Manolis Koubarakis (Technical University of Crete, Greece) Vana Kalogeraki (University of California, Riverside, USA)
Panel Chair	Aris Ouksel (University of Illinois, Chicago, USA)
Organization Chair	Klemens Böhm (University of Magdeburg, Germany)

Steering Committee

Karl Aberer (EPFL, Lausanne, Switzerland)
Sonia Bergamaschi (University of Modena and Reggio-Emilia, Italy)
Manolis Koubarakis (Technical University of Crete)
Paul Marrow (BTexact Technologies,UK)
Gianluca Moro (University of Bologna, Cesena, Italy)
Aris M. Ouksel (University of Illinois, Chicago, USA)
Claudio Sartori (CNR-CSITE, University of Bologna, Italy)
Munindar P. Singh (North Carolina State University, USA)

Program Committee

Ozalp Babaoglu, University of Bologna, Italy
Klemens Böhm, University of Magdeburg, Germany
Beng Chin Ooi, National University of Singapore, Singapore
Partha Dasgupta, Arizona State University, USA
Alex Delis, Polytechnic University, USA
Fausto Giunchiglia, University of Trento, Italy
Zachary G. Ives, University of Washington, USA
Carole Goble, University of Manchester, UK
Oliver Guenther, Humboldt University, Germany
Dimitris Gunopoulos, University of California at Riverside, USA
Manfred Hauswirth, EPFL, Switzerland
Achilles D. Kameas, Computer Technology Institute, Greece
Yannis Labrou, Fujitsu Labs of America, USA
Witold Litwin, University of Paris 6, France
Ling Liu, Georgia Institute of Technology, USA
Peri Loucopoulos, UMIST, Manchester, UK
Dejan Milojicic, Hewlett-Packard Labs, USA
Alberto Montresor, University of Bologna, Italy
Jean-Henry Morin, University of Geneva, Switzerland
John Mylopoulos, University of Toronto, Canada
Wolfgang Nejdl, Learning Lab Lower Saxony, Germany
Dimitris Papadias, Hong Kong University of Science and Technology, China
Mike Papazoglou, Tilburg University, Netherlands
Evaggelia Pitoura, University of Ioannina, Greece
Dimitris Plexousakis, Institute of Computer Science, FORTH, Greece
Onn Shehory, IBM Haifa, Israel
Spiros Skiadopoulos, National Technical University of Athens, Greece
Katia Sycara, Robotics Institute, Carnegie Mellon University, USA
Peter Triantafillou, University of Patras, Greece
Martin Wolpers, Learning Lab Lower Saxony, Germany

Referees

W. Black
Claus Boyens
David Buttler
James Caverlee
Eleni Christopoulou
Matthias Fischmann
Christos Goumopoulos
Tasos Gounaris
Verena Kantere
Mujtaba Khambatti
Georgia Koloniari
Bin Liu
Nikolaos Ntarmos
Themis Palpanas
Michel Pawlak
I. Petrounias
Theoni Pitoura
Th. Schwarz

Table of Contents

Invited Talk

Structure in P2P Networks

Semantics and Data Integration

Data Streams and Publish/Subscribe

Data Structures and Query Processing

Design Issues and Challenges for RDF- and Schema-Based Peer-to-Peer Systems

Wolfgang Nejdl

Learning Lab Lower Saxony and University of Hannover, 30539 Hannover
nejdl@learninglab.de

Abstract. Databases have employed a schema-based approach to store and retrieve structured data for decades. For peer-to-peer (P2P) networks, similar approaches are just beginning to emerge. While quite a few database techniques can be re-used in this new context, a P2P data management infrastructure poses additional challenges which have to be solved before schema-based P2P networks become as common as schema-based databases. We will describe some of these challenges and discuss approaches to solve them, basing our discussion on the design decisions we have employed in our Edutella infrastructure, a schema-based P2P network based on RDF and RDF schemas.

K. Aberer et al. (Eds.): VLDB 2003 Ws DBISP2P, LNCS 2944, p. 1, 2004.

SIL: Modeling and Measuring Scalable Peer-to-Peer Search Networks*

Brian F. Cooper and Hector Garcia-Molina

Department of Computer Science
Stanford University
Stanford, CA 94305 USA
{cooperb,hector}@db.Stanford.EDU

Abstract. The popularity of peer-to-peer search networks continues to grow, even as the limitations to the scalability of existing systems become apparent. We propose a simple model for search networks, called the *Search/Index Links* (SIL) model. The SIL model describes existing networks while also yielding organizations not previously studied. Using analytical and simulation results, we argue that one new organization, *parallel search clusters*, is superior to existing supernode networks in many cases.

1 Introduction

Peer-to-peer search networks have become very popular as a way to effectively search huge, distributed data repositories. On a typical day, systems such as Kazaa support several million simultaneous users, allowing them to search hundreds of millions of digital objects totaling multiple petabytes of data. These search networks take advantage of the large aggregate processing power of many hosts, while leveraging the distributed nature of the system to enhance robustness. Despite the popularity of peer-to-peer search networks, they still suffer from many problems: nodes quickly become overloaded as the network grows, and users can become frustrated with long search latencies or service degradation due to node failures. These issues limit the usefulness of existing peer-to-peer networks for new data management applications beyond multimedia file sharing.

We wish to develop techniques for improving the efficiency and fault tolerance of search in networks of autonomous data repositories. Our approach is to study how we can place indexes in a peer-to-peer network to reduce system load by avoiding the need to query all nodes. The scale and dynamism of the system, as large numbers of nodes constantly join and leave, requires us to re-examine index replication and query forwarding techniques.

However, the space of options to consider is complex and difficult to analyze, given the bewildering array of options for search network topologies, query routing and processing techniques, index and content replication, and so on. In

* This material is based upon work supported by the National Science Foundation under Award 9811992.

K. Aberer et al. (Eds.): VLDB 2003 Ws DBISP2P, LNCS 2944, pp. 2–16, 2004.

order to make our exploration more manageable, we separate the process into two phases. In the first phase, we construct a coarse-grained *architectural model* that describes the topology of the connections between distributed nodes, and models the basic query flow properties and index placement strategies within this topology. In the second phase, we use the insights gained from the architectural model to develop a finer-grained *operational model*, which describes at a lower level the actual processing in the system. The operational model allows us to study alternatives for building and maintaining the topology as nodes join and leave, directing queries to nodes (for example, using flooding, random walks or routing indices), parallel versus sequential query submission to different parts of the network, and so on.

Our focus in this paper is on the first phase architectural model. We have developed the Search/Index Link (SIL) model for representing and visualizing peer-to-peer search networks at the architectural level. The SIL model helps us to understand the inherent properties of many existing network architectures, and to design and evaluate novel architectures that are more robust and efficient. Once we understand which architectures are promising, ongoing work can examine operational issues. For example, in [4], we examine the operational question of how the architectures described here might be constructed. In this paper, we first present and analyze the SIL model, and show how it can lead to new search network architectures. Then, using analytical and simulation results, we show that our new organizations can be superior to existing P2P networks in several important cases, in terms of both efficiency and fault tolerance.

2 The Search/Index Link Model

A *peer-to-peer search network* is a set of peers that store, search for, and transfer digital documents. We consider here content-based searches, such as keyword searches, metadata searches, and so on. This distinguishes a peer-to-peer search network from a distributed hash table [14,11], where queries are to locate a specific document with a specific identifier (see Section 5 for more discussion about SIL versus DHTs). Each peer in the network maintains an index over its content (such as an inverted list of the words in each document) to assist in processing searches. We assume that the index is sufficient to answer searches, even though it does not contain the whole content of the indexed documents.

The search network forms an *overlay* on top of a fully-connected underlying network infrastructure. The topology of the overlay determines where indexes are placed in the network, and how queries reach either a data repository or an index over that repository's content. Peers that are neighbors in the overlay are connected by network links that are logically persistent, though they may be implemented as connection-oriented or connectionless.

The Search/Index Link (SIL) model allows us to describe and visualize the overlay topology. In the SIL model, there are four kinds of network links, distinguished by the types of messages that are sent, and whether a peer receiving a message forwards the message after processing it:

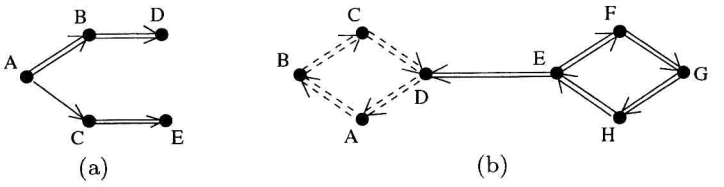

Fig. 1. Networks: (a) with search links, (b) with search and index links.

- A *non-forwarding search link* (NSL) carries search messages a single hop in the overlay from their origin. For example, a search generated at one peer A will be sent to another peer B, but not forwarded beyond B. Peer B processes each search message and returns results to A.
- A *forwarding search link* (FSL) carries search messages from A to B. Peer B will process each search message, return search results to A, and forward the message along any other forwarding search links originating at B. If A is not the originator of the query, it should forward any search results received from B (and any other nodes) along the FSL on which A received the query.
- A *non-forwarding index link* (NIL) carries index update messages one hop in the overlay from their origin. That is, updates occurring at A will be sent to B, but not forwarded. Peer B adds A's index entries to its own index, and then effectively has a copy of A's index. Peer B need not have a full copy of A's content.
- A *forwarding index link* (FIL) carries index update messages from A to B, as with non-forwarding index links, but then B forwards the update message along any other forwarding index links originating at B.

For FSLs and FILs, messages should have unique ids, and a peer should discard duplicate messages without processing or forwarding them. This avoids infinite propagation of messages if the network has cycles.

Network links are directed communications channels. A link from peer A to peer B indicates that A sends messages to B, but B only sends messages to A if there is also a separate link from B to A. Modeling links as directed channels makes the model more general. An undirected channel can be modeled as a pair of directed links going in opposite directions. For example, the links in Gnutella can be modeled as a pair of FSLs, one in each direction. Although forwarding links may at first glance seem more useful, we will see how non-forwarding links can be used in Section 3.

Figure 1a shows an example network containing search links. Non-forwarding search links are represented as single arrows ($\longrightarrow$) while forwarding search links are represented as double arrows ($\Longrightarrow$). Imagine that a user submits a query to peer A. Peer A will first process the query and return any search results it finds to the user. Node A will then send this query to both B and C, who will also process the query. Node B will forward the query to D. Node C will not forward

the query, since it received the query along an NSL. The user's query will not reach E at all, and E's content will not be searched for this query.

A peer uses an *index link* to send copies of index entries to its neighbors. These index entries allow the content to be searched by the neighbors without the neighbors having to store the peer's actual content. For example, consider a peer A that has an index link to a peer B. When B processes a query, it will return search results both for its own content as well as for the content stored at A. Peer A need not process the query at all. We say that B is *searched directly* in this case, while A is *searched indirectly*.

Whenever a peer creates a new index entry or modifies an existing entry, it should send a message indicating the change along all of its outgoing index links. A peer might create an index over all of its locally stored documents when it first starts up, and should send all of the index entries to each of its index link neighbors. Similarly, if a node deletes a document, it would remove the corresponding entries from its own index as well as notifying its index link neighbors to do the same.

Figure 1b shows a network that contains both search and index links. Index links are represented as dashed lines, single (⇢) for non-forwarding index links and double (⇛) for forwarding index links. (Note that Figure 1b contains only FILs.) Nodes A, B, C and D are connected by a "ring" of FILs. An index update occurring at peer A will thus be forwarded to B, C, D and back to A (A will not forward the update again). In fact, all four of the nodes $A...D$ will have complete copies of the indexes at the other three nodes in the index "ring". Nodes E, F, G and H are connected by FSLs, and a search originating at any peer $E...H$ will reach, and be processed by, the three other nodes on the search "ring." Notice that there is also an FSL between E and D. Any query that is processed by E will be forwarded to D, who will also process the query. Since D has a copy of the indexes from $A...C$, this means that any query generated at E, F, G and H will effectively search the content of all eight nodes in the network. In contrast, a query generated at nodes $A...D$ will be processed at the node generating the query, and will only search the indexes of the nodes $A...D$.

A search path from X to Y indicates that queries submitted to X will eventually be forwarded to Y, either through a sequence of FSLs or by a single NSL. For example, in Figure 1b there is a search path from F to D but not from D to F. There is (trivially) a search path from a node to itself. Similarly, an *index path* from X to Y is a sequence of FILs from X to Y, or one NIL from X to Y. In this case, X's index updates will be sent to Y, and Y will have a copy of X's index.

2.1 "Good" Networks

The network links we have discussed above are not by themselves new. Forwarding search links are present in Gnutella, forwarding index links are used in publish/subscribe systems, non-forwarding index links are used in supernode networks, and so on. However, different link types tend to be used in isolation or for narrowly specific purposes, and are rarely combined into a single, general

model. Our graphical representation allows us to consider new combinations. In fact, the number of search networks of n nodes that can be constructed under the SIL model is exponential in n^2. Only a small fraction of these networks will allow users to search the content of most or all the peers in the network, and an even smaller fraction will also have desirable scalability, efficiency or fault tolerance properties. We want to use the SIL model to find and study "good" networks, and this of course requires defining what we mean by "good."

First, we observe that a search network only meets users' needs if it allows them to find content. Since content may be located anywhere in the network, a user must be able to effectively search as many content repositories as possible, either directly or indirectly. We can quantify this goal by defining the concept of *coverage*: the fraction of peers in the network that can be searched, either directly or indirectly, by a query generated by a peer p. Ideal networks would have *full coverage*: all peers have *coverage* $= 1$, and if content exists anywhere in the network, users can find it. It may be necessary to reduce coverage in order to improve network efficiency.

Even a network that has full coverage may not necessarily be "good." Good networks should also be efficient, in the sense that peers are not overloaded with work answering queries. One important way to improve the efficiency of a network is to reduce or eliminate redundant work. If peers are duplicating each other's processing, then they are doing unnecessary work.

Definition 1. *A search network N has* redundancy *if there exists a network link in N that can be removed without reducing the coverage for any peer.*

Intuitively, redundancy results in messages being sent to and processed by peers, even when such processing does not add to the network's ability to answer queries.

Redundancy can manifest in search networks in four ways:

- *Search/search redundancy* occurs when the same peer P processes the same query from the same user multiple times.
- *Update/update redundancy* occurs when the same peer P processes the same update multiple times.
- *Search/index redundancy* means a peer A processes a query even though another peer B has a copy of A's index and processes the same query.
- *Index/index redundancy* is where two different peers B and C both process a search over a copy of a third peer A's index.

In each of these cases, a node is doing work that is unnecessary to achieve high or full coverage.

Note that redundancy may actually be useful to improve the fault tolerance of the system, since if one node fails another can perform its processing. Moreover, redundancy may be useful to reduce the time a user must wait for search results, if a node near the user can process the user's search even when this processing is redundant. However, fault tolerance and search latency tradeoff with efficiency, since redundancy results in extra work for peers.

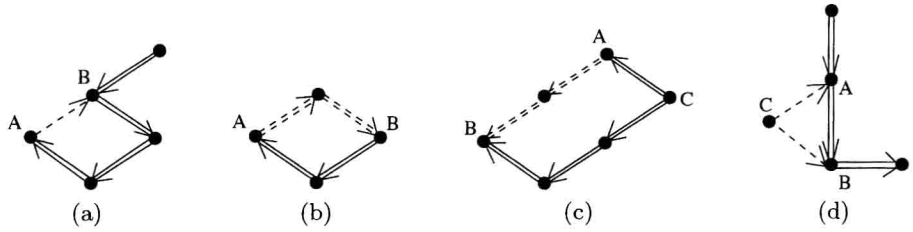

Fig. 2. Networks: a. with search/index redundancy, b. no search/index redundancy, c. search-fork, and d. index-fork

2.2 Topological Features of Networks with Redundancy

The concept of "redundancy" and even the subconcepts like search/index redundancy are quite general. Rather than avoiding generalized redundancy when designing a peer-to-peer search network, it is easier to identify specific features of network topologies that lead to redundancy, and avoid those features.

A feature that causes search/index redundancy is a specific type of cycle called a *one-cycle*. A *one-index-cycle* is a version of a one-cycle:

- A *one-index-cycle* is when a node A has an index link to another node B, and B has a search path to A.

An example is shown in Figure 2a. This construct leads to redundant processing, since B will answer queries over A's index, and yet these queries will be forwarded to A who will also answer them over A's index. More formally, a one-index-cycle fits our definition of *redundancy* because at least one link in the cycle can be removed without affecting coverage: the index link from A to B. Another version of a one-cycle is a *one-search-cycle*:

- A *one-search-cycle* is when a node A has an search link to another node B, and B has an index path to A.

While *one-cycles* (one-index-cycles and one-search-cycles) cause redundancy, not all cycles do. Consider the cycle in Figure 2b. This cycle may seem to introduce redundancy in the same way as a one-cycle, except that none of the links can be removed without reducing coverage for some node.

Another feature that causes search/index redundancy is a *fork*.

- A *search-fork* is when a node C has a search link to A and a search path to B that does not include A, and there is an index path from A to B.

An example of a search-fork is shown in Figure 2c. Again, A will process any searches from C unnecessarily, since B can process the queries for A. The redundant link in this example is the link $C \Longrightarrow A$. We specify that there is a search path from C to B that does not include A because if the only path from C to B included A there would be no link that could be removed without reducing coverage. The analog of a search-fork is an *index-fork*; an example is shown in Figure 2d.

- An *index-fork* is when a node C has an index link to A and an index path to B that does not include A, and there is a search path from A to B.

The third feature that causes redundancy is a *loop*:

- A *search-loop* is when a node A has an search link l_s to another node B, and also another search path to B that does not include l_s.
- An *index-loop* is when a node A has an index link l_i to another node B, and also another index path to B that does not include l_i.

Avoiding all of these topological features is sufficient to avoid the general property of redundancy in a network.

Theorem 1. *If a network has no one-cycles, forks or loops, then it has no redundancy.*

Proof. The proof is straightforward: a redundant edge implies one of the named features. The full proof is available in the extended version of this paper [5]. □

3 Network Archetypes

We can now identify some archetypical network organizations described by the SIL model. Each archetype is a family of topologies that share a common general architecture. We restrict our attention to somewhat idealized networks, that is, non-redundant networks with full coverage, in order to understand the inherent advantages and disadvantages of various architectures. We do not claim to examine the entire design space of peer-to-peer topologies. Instead, by looking at some representative archetypes of a particular design point, that is, non-redundant full-coverage networks, we can both understand that design point clearly and also illustrate the value of SIL as a design tool.

We consider only the static topologies described by the SIL architectural model, in order to determine which topologies have efficiency or fault tolerance benefits and are worth examining further. If a particular archetype is selected for a given application, there are then operational decisions that must be made. For example, if a supernode archetype (described fully below) is chosen as desirable, there must be a way to form peers into a supernode topology as they join the system. One way to form such a network is to use a central coordinator that selects which nodes are supernodes and assigns them responsibility for non-supernodes. Alternatively, nodes could decide on their own whether to be supernodes or not, and then advertise their supernode status to connect to other, non-supernode peers. This dynamic process of forming a specific topology is outside the scope of this paper, as we wish to focus for now on which topology archetype is most desirable under various circumstances. For a discussion on how a topology can be constructed dynamically, see [4,16].

Also, we focus on networks with no search/index or index/index redundancy. The impact of search/search and update/update redundancies is mitigated by the fact that a node processes only one copy of a duplicate search or update message and discards the rest (see Section 2). In contrast, search/index and