

# Contents

	<b>Preface</b> .....	<b>xi</b>
<b>1</b>	<b>Introduction</b> .....	<b>1</b>
1.1	Peer-to-Peer Systems .....	1
1.1.1	Concepts .....	2
1.1.2	Architectures .....	3
1.2	Peer-to-Peer Data Management .....	11
1.2.1	Architectures .....	13
1.2.2	Design Dimensions .....	15
1.2.3	Overview of the Lecture .....	17
<b>2</b>	<b>Structured Peer-to-Peer Databases</b> .....	<b>19</b>
2.1	Range Partitioning .....	19
2.1.1	Static Range Partitioning .....	21
2.1.2	Dynamic Range Partitioning .....	22
2.1.3	Load Balancing .....	23
2.1.4	Overlay Network Construction .....	28
2.1.5	Efficient Range Query Processing .....	33
2.2	Complex Query Processing .....	35
2.2.1	Managing Multiple Attributes .....	36
2.2.2	Join Queries .....	38
2.2.3	Aggregation Queries .....	40
2.3	Web Data Management .....	50
2.3.1	Triple Indexing .....	51
2.3.2	Schema-based Indexing .....	53
<b>3</b>	<b>Peer-to-peer Data Integration</b> .....	<b>55</b>
3.1	Peer-to-peer Data Integration .....	55
3.2	Query Rewriting .....	57
3.2.1	Attribute Mapping .....	57
3.2.2	Mapping Tables .....	58
3.3	View-based Data Integration .....	61

3.3.1	Global-as-View Mappings	61
3.3.2	Global-and-Local as View Mappings	63
3.4	Incomplete and Incorrect Mappings	67
3.4.1	Handling Incomplete Mappings	68
3.4.2	Discovering Mappings	68
3.4.3	Correcting Mappings	70
<b>4</b>	<b>Peer-to-peer Retrieval</b>	<b>75</b>
4.1	Full Text Search	75
4.1.1	Bloom Filters	80
4.1.2	Shortening of Posting Lists	81
4.1.3	Caching	83
4.1.4	Fagin's Algorithm	84
4.1.5	Peer Summaries	86
4.2	Structured Document Search	87
4.2.1	Path Indexing	88
4.2.2	Filtering	92
<b>5</b>	<b>Semantic Overlay Networks</b>	<b>95</b>
5.1	Concept Spaces	95
5.2	Semantic Overlay Network Construction	99
5.3	Protocol-driven SON	100
5.4	Mapping-based SON	103
5.4.1	Mapping Categories	103
5.4.2	Mapping Concept Vectors	104
<b>6</b>	<b>Conclusion</b>	<b>109</b>
<b>A</b>	<b>Notational Conventions</b>	<b>113</b>
	<b>Bibliography</b>	<b>117</b>
	<b>Author's Biography</b>	<b>133</b>
	<b>Index</b>	<b>135</b>

# Preface

Peer-to-peer data management connects the fields of peer-to-peer systems and data management. The nature of this connection is twofold. In peer-to-peer systems, search over large data sets is a key functionality; therefore, they have a natural link to the area of data management. The basic concepts for realizing scalable applications on the Internet that are at the core of peer-to-peer technology, on the other hand, inspired also novel large-scale data management architectures.

The focus of this lecture will be on the representation and search of structured data and documents in peer-to-peer data management systems. We adopt a wide perspective on the notions of data and search, including multiple facets of information management such as data integration, document management and text retrieval. This is motivated by the observation that these are key areas for which peer-to-peer architectures have been typically considered. In order to limit the scope of the lecture, we do not cover aspects related to state management and data dissemination, such as updates, transactions, data stream management, continuous querying and publish-subscribe. These areas are less mature today and will be shortly surveyed at the end of the lecture.

We will assume that the reader is familiar with the fundamentals of both the areas of peer-to-peer systems and data management. For peer-to-peer systems, there exist numerous surveys and for data management there exists a rich source of comprehensive textbooks. We will, in this book, recall from both fields the more specialized key concepts that will be necessary for the understanding of the specific techniques we discuss. These will be introduced in separate information boxes.

The lecture will cover four different types of peer-to-peer data management systems that are characterized by the type of data they manage and the search capabilities they support. The first type are *structured peer-to-peer data management systems* which support structured query capabilities for standard data models, in particular, relational data and Web data represented in RDF. The second type are *peer-to-peer data integration systems*. These support querying of databases that are using heterogeneous database schemas without requiring a common global schema. The third type are *peer-to-peer document retrieval systems* that enable document search based both on the textual content and the document structure, e.g., represented as XML. Finally, we will introduce *semantic overlay networks*, which in a sense generalize the previous types of systems. They manage data from both hierarchically organized and multi-dimensional semantic spaces that are equipped with a similarity measure to model semantic proximity.

Karl Aberer  
May 2011



## CHAPTER 1

# Introduction

Peer-to-peer systems have generated a wide interest over the last decade, both in research and practice. Sharing of media content, in particular of music and video, through peer-to-peer systems has become a highly popular and the most bandwidth consuming application on the Internet. It has also induced huge debates on the legality of this activity and the economic model employed by the media industry. From a data management perspective, peer-to-peer content sharing systems pose interesting challenges, as they can be understood as very large scale and widely distributed databases. For example, the problem of searching for content based on a variety of criteria within a highly decentralized system poses interesting challenges that had not been addressed at that time. Initial practical solutions were based on simplistic albeit robust approaches, such as the famous Gnutella protocol. In parallel intensive research efforts in distributed systems, networking and data management started with the goal of designing more efficient solutions to accessing and managing data in a peer-to-peer architecture. This resulted in a rich set of novel algorithms and system architectures. Many of these address classical data management problems under a completely novel perspective. Scalability, minimizing communication cost and decentralized control are among the most prominent aspects that motivated this research. Over time, the focus expanded from the initial problem of content search, to many data management problems that can be studied in a peer-to-peer architecture. In this lecture, we will provide an overview of the most important principles, solutions and applications based on the peer-to-peer paradigm for data management problems.

In this chapter, we will first survey the basic notions of peer-to-peer systems on which we will rely in this lecture. Then we will introduce a taxonomy of peer-to-peer data management architectures to set the framework for the subsequent chapters in this lecture.

## 1.1 PEER-TO-PEER SYSTEMS

The notion of peer-to-peer systems is used in a wide variety of contexts and implies different technical concepts and approaches. In the following, we introduce a taxonomy of the key concepts that are relevant for the purpose of this book. We will also review some key technical approaches to peer-to-peer system architectures that peer-to-peer data management systems rely on. For more detailed technical discussions of peer-to-peer technologies that have emerged over the past 10 years, we refer the reader to one of the numerous surveys and books [[Aberer and Hauswirth, 2002](#), [Aberer et al., 2005](#), [Androutsellis-Theotokis and Spinellis, 2004](#), [Korzun and Gurtov, 2010](#), [Lua et al., 2005](#), [Milojicic et al., 2002](#), [Oram, 2001](#), [Risson and Moors, 2006](#), [Vu et al., 2010](#)].

## 2 1. INTRODUCTION

### 1.1.1 CONCEPTS

In its most general sense, a *peer-to-peer system* is a computational system consisting of a set of *peers*  $P$  that jointly perform a common task and are connected to *neighbors* through *links* in a *logical network*. This logical network is called the *peer-to-peer overlay network*, or sometimes short peer-to-peer network, overlay network or simply network, when the context is clear. The peers communicate to their neighbors through a *communication network*, which is called the *underlay network*. In the literature, peers are also frequently called nodes or agents, but for the sake of consistency, in this book, we will always use the term peer. Links are often called references, connections or vertices, but we will consistently use the term link. The communication network is typically the Internet, but other means of communication like wireless communication networks may be considered. Even communication among peers that are hosted on a common server might be considered as a realization of the underlay network.

In the ideal case, different peers are assumed to have *equivalent roles*, state is distributed among the peers and control is *decentralized*. In other words, there exists no entity that has access to the global state of the system or exerts global control. As a result, decisions are always based on *local information* of peers, and the global system behavior is an *emergent property* of the system. Peer-to-peer systems are also often considered as *self-organizing* systems. The nature and formalization of the concepts of *emergence* and self-organization is an intensely debated topic into which we will not enter here [Serugendo et al., 2004].

The first essential function of a peer-to-peer system is *overlay network maintenance*. Every peer  $p \in P$  maintains a set of neighbors  $N(p) \subseteq P$  of which it knows their physical address in the underlay network. The data structure used to store the neighbors, their physical addresses and potentially other information about the neighbors is called the peer's *routing table*. Different overlay network architectures differ in the structure and contents of their routing tables. A characteristic property of peer-to-peer networks is the small size of the set of neighbors as compared to the *network size*  $n = |P|$ , typically constant or logarithmic in  $n$ . This property implies that the state of a peer-to-peer network is necessarily distributed since no peer has a global view of the peer-to-peer network.

Peers can autonomously decide to join or leave a peer-to-peer network. The process of joining and leaving is called *churn*. For joining a network a peer  $p \in P$  needs to contact a peer  $p_c \in P$  that is currently part of the network. Therefore, the join operation has the signature  $join(p, p_c)$ . Peers can always autonomously leave the network; therefore, the leave operation is of the form  $leave(p)$ . Overlay network maintenance protocols ensure that under churn the overlay network structure is maintained, i.e., that peers have routing tables that satisfy the constraints imposed by the overlay network architecture. The overlay network maintenance protocols are distributed, thus resulting in decentralized control of the peer-to-peer overlay network.

Peer-to-peer systems have been designed to perform a variety of tasks. However, most frequently, they are used to share and provide access to a set of *resources*  $R$ . These resources can be information resources such as documents, media contents, data, knowledge or services, or physi-

cal resources. Examples of physical resources are storage space, computational capacity or network bandwidth. We will call such systems *peer-to-peer resource sharing systems*.

Resources in a peer-to-peer system are either identified by a unique *resource identifier* from a *resource identifier space*  $I(R)$  or by *resource properties* represented in application-specific data structures. A peer-to-peer resource sharing system supports basic functions for *resource management*. Basic examples of such functions are the following:

- $insert(p, r)$  : add a resource  $r \in R$  at peer  $p \in P$ .
- $lookup(p, id(r)) : p_r$  : request from a peer  $p \in P$  the physical address of a peer  $p_r \in P$  that holds a resource with identifier  $id(r) \in I(R)$ .
- $query(p, q) : P_q$  : issue a query  $q$  at peer  $p \in P$  returning the physical addresses of a set of peers  $P_q \subseteq P$  holding resources that satisfy the conditions of the query.

The access to the resources at a peer is realized through application-specific protocols. In order to implement these functions, the peer-to-peer networks use *routing protocols* for locating peers that hold specific resources. Routing protocols are distributed algorithms that use the overlay network structure to exchange messages among peers. The overlay network structure together with the routing protocols are the essential constituents of a peer-to-peer system architecture.

A special type of resource are the peers themselves. They are identified by *peer identifiers* from a peer identifier space  $I(P)$ . The same set of operations as provided for general resources can be applied to peers. For example, the lookup operation would have the form  $lookup(p, id(p_s)) : p_s$  and return the physical address of a peer given its identifier. Figure 1.1 illustrates the concepts introduced.

### 1.1.2 ARCHITECTURES

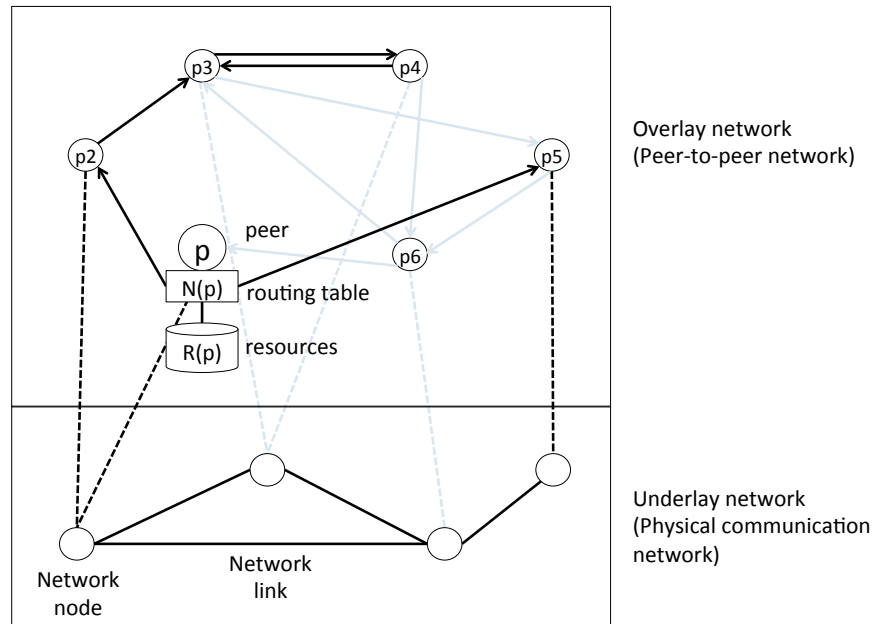
Peer-to-peer overlay networks are the essential constituent of a peer-to-peer system. Any function of the peer-to-peer system implemented as distributed algorithm makes use of the overlay network, and any state information that is present in a peer-to-peer system is accessed through the overlay network. There exist three main types of overlay network architectures:

- Unstructured overlay networks
- Structured overlay networks
- Hierarchical overlay networks

We introduce in the following the main principles for each of these architectures and provide a canonical example for each type.

**Unstructured overlay networks.** In an unstructured overlay network, the choice of neighbors of a peer is not constrained by the state of the peers. In other words, peers can freely connect to any other peer in the network, by obtaining its physical network address. Similarly, also the choice of

#### 4 1. INTRODUCTION



**Figure 1.1:** This figure summarizes some essential aspects of a peer-to-peer network. The peers in the peer-to-peer network are connected by logical links. Each peer, such as peer  $p$ , has a routing table with information on the peer's neighbors and resources that it manages. A logical link may correspond to a path in the underlying communication network. For example, the link between peer  $p$  and peer  $p_5$  corresponds to two physical hops in the communication network. Conversely, peers that are separated through multiple links in the overlay network, may be physically close, e.g., peers  $p$  and  $p_3$ . Different peers may be hosted in the same physical node, for example, peers  $p$  and  $p_2$ . The overlay network may contain cycles, e.g., between peers  $p_3$  and  $p_4$ .

which resources a peer can manage is not constrained by the state of the peer. In a pure unstructured overlay network, a peer does not maintain any information about the state of other peers in its routing tables. However, this assumption is frequently relaxed for optimizing the performance of unstructured overlay networks. As a result of these architectural principles, when searching a resource in the overlay network, typically, a large fraction or the whole network needs to be accessed.

**Gnutella.** Gnutella was the first unstructured peer-to-peer system used in practice and the starting point for a family of peer-to-peer protocols based on *gossiping* [Ripeanu, 2001]. The network structure consists of a directed graph with constant out-degree. For searching, a *breadth-first search strategy (BFS)* is adopted: a query message is forwarded to all neighbors, who, in turn, forward it to their neighbors till the message has been forwarded a *time-to life (TTL)* number of steps. Peers remember the recently sent messages to avoid cyclic forwarding. If a result is found, a query reply message is sent back to the query originator along the same path the query message has been forwarded. Network maintenance relies on a similar approach. A peer that is joining the network or in need of new neighbors announces its presence by broadcasting a *ping message* to the network, using the BFS strategy. Any peer receiving a ping message may decide to respond with a *pong message* that is routed back along the path the ping message arrived. Several improvements have been proposed to decrease the high message traffic incurred by the BFS protocol [Lv et al., 2002, Tsoumakos and Roussopoulos, 2006].

- *Selective forwarding.* A search message is only forwarded to a fraction of the known neighbors. If this fraction is chosen carefully, still the whole network can be reached. This approach is called *percolation search* [Sarshar et al., 2004].
- *Iterative deepening.* The search starts with a low TTL. Only if no results are found, a new search with higher TTL is initiated. This approach reduces message traffic in particular when searching frequently occurring resources.
- *Random walkers.* Instead of relying on breadth-first search, this strategy uses *depth-first search*. A random walker selects among the neighbors uniformly at random one peer to forward the query message. In this approach, the TTL is chosen significantly higher. One or more parallel random walkers can be used.

The main characteristics of unstructured overlay networks can be summarized as follows. They typically incur high message traffic for performing searches, while the maintenance of the overlay network and the update of resources is less costly. They are robust against failures since for their operation no specific constraints, apart from avoiding disconnection, need to be satisfied. Their search mechanism does not limit the expressivity of queries used, as queries are evaluated locally at each peer.

**Structured overlay networks.** In this architecture, the structure of the overlay network as well as the assignment of responsibility of peers to manage specific resources follows a global organization principle. The organization of a structured overlay network is based on the association of the peers and the resources with some common, application-specific identifier space  $I$ , in other terms,  $I = I(P) = I(R)$ . Domain-specific identifiers of resources and peers are mapped to the identifier space

## 6 1. INTRODUCTION

using hash functions  $h_R : R \rightarrow I$  and  $h_P : P \rightarrow I$ . For resources the domain-specific identifiers consist typically of some metadata, whereas for peers they are peers' IP addresses.

The identifier space  $I$  is equipped with a topology; typically, it is a one or multi-dimensional metric space. Both, the selection of neighbors of a peer and the responsibility to manage certain resources, depend on the peer identifiers. The set of resource identifiers a peer is responsible for is defined as a local neighborhood of the peer identifier. The routing tables consist usually of *short-range links* to direct neighbors in the peer identifier space, which ensure basic connectivity of the network, and *long-range links* to remote peers, which accelerate resource lookup by resource identifier, when those resources are found in a distant region of the identifier space. The significant difference to unstructured overlay networks is that lookups are performed by *greedy routing*. A lookup message, searching for a resource identifier, is always forwarded to the neighbor that reduces the distance to the target the most. Thus, searches do not flood the whole network. On the other hand, maintenance of a structured overlay network under churn may become costly since the structural constraints need to be maintained.

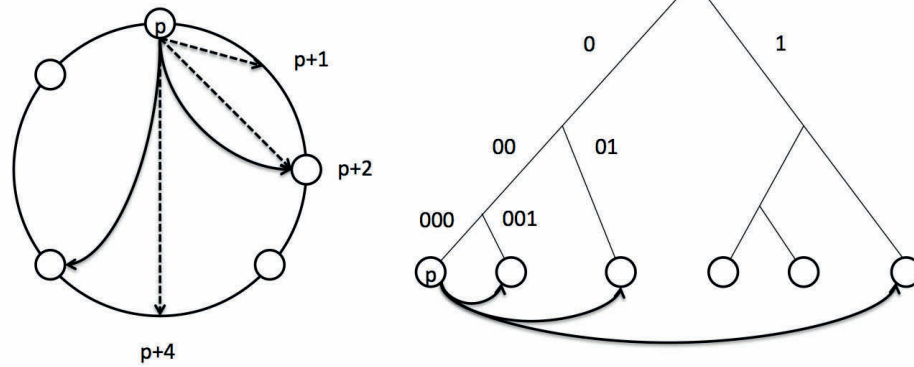
Numerous structured overlay network approaches have been proposed and studied over the past years. Here we distinguish these approaches primarily by the type of identifier space they employ and mention some of the well-known, canonical approaches.

- *One-dimensional ring*. The identifiers are binary strings of fixed length  $l$ . They are arranged in a ring topology with distance metrics  $d(id_1, id_2) = |id_2 - id_1| \bmod 2^l, id_1, id_2 \in I$ . The most prominent representative of this approach is *Chord* [Stoica et al., 2001].
- *Binary prefix trees*. The identifiers are taken from the set of bitstrings  $\{0, 1\}^*$ . The distance metrics can be given by the XOR-metrics, i.e.,  $d(id_1, id_2) = id_1 \oplus id_2, id_1, id_2 \in I$ , where  $(id_1 \oplus id_2)_i = 1$  if  $(id_1)_i \neq (id_2)_i$  and 0, otherwise. The most prominent representatives are P-Grid [Aberer, 2001] and Kademia [Maymounkov and Mazières, 2002].
- *m-dimensional torus*. The identifiers are taken from an  $m$ -dimensional torus, typically  $[0, 1]^m$ . The metrics is the Euclidean distance. The most prominent representative of this approach is CAN [Ratnasamy et al., 2001].

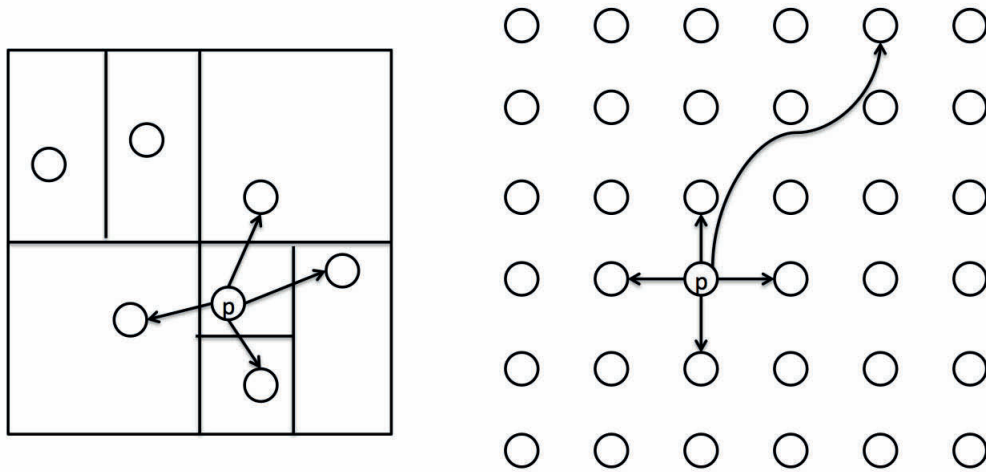
For an extensive and detailed review of structured overlay network approaches, we refer in particular to Korzun and Gurtov [2010].

**Chord.** Chord has been among the first proposals of a *distributed hash table* [Stoica et al., 2001]. It is based on *consistent hashing*, which applies a uniform hash function, such as SHA-1, to map resources and peers into a common identifier space. The identifier space in Chord consists of fixed-length binary strings of length  $l$  on a one-dimensional ring. The distance on the ring is computed modulo  $2^l$ . By virtue of its hashed identifier, a peer  $p$  becomes responsible to manage the resources that are mapped into the interval  $[p, \text{successor}(p))$ , where  $\text{successor}(p)$  is the peer with the next higher identifier on the ring. Peers are connected to their immediate successor which provides basic routing capability in the overlay network. In addition, peers maintain long-range links in exponentially increasing distances. More precisely, Chord requires a peer  $p$  to maintain links to the peers  $\text{successor}(p + 2^i - 1)$ ,  $i = 1, \dots, l$ . As a result greedy search succeeds in  $O(\log n)$  steps. For maintenance, peers that join have to obtain their long-range links and at the same time other peers may have to update their routing tables as a result of the join operation. The cost for a join operation is  $O(\log^2 n)$ . Chord has been the basis for numerous structured overlay network designs based on variations of its principles [Korzun and Gurtov, 2010].

**P-Grid.** P-Grid was an early proposal of a structured overlay network for supporting data-oriented applications [Aberer, 2001, Aberer et al., 2002b]. P-Grid uses an order-preserving hash function to map data into variable-length binary strings. Peers are logically organized in a binary tree. A peer with identifier  $p$  corresponds to the leaf node of the tree, which is reached by the binary path  $p$ . In its routing table, it maintains links to peers that correspond at each level of the tree to the opposite half of the subtree the peer belongs to. Formally, a peer with identifier  $p = b_1 \dots b_l$  maintains for each prefix  $b_1 \dots b_{l'}$ ,  $l' < l$  a link to a peer with identifier prefix  $b_1 \dots (1 - b_{l'})$ . Searching for a key, it then proceeds by identifying in the routing table the peer with the longest shared prefix and forwarding the search request to it. For balanced trees, this implies by construction that the search cost is bounded by  $\log_2 n$ , and it has been shown that, also, for unbalanced trees, the search cost is  $O(\log n)$ . Similarly as for Chord, a peer joining can build up its routing table in  $O(\log^2 n)$ . Other approaches based on prefix routing have been Kademlia [Maymounkov and Mazières, 2002], which introduced the concept of XOR metric, and Pastry [Rowstron and Druschel, 2001] and Tapestry [Zhao et al., 2004], which use higher-degree trees as underlying abstraction.



**Figure 1.2:** Illustration of the overlay network structure of Chord and P-Grid.



**Figure 1.3:** Illustration of the overlay network structure of CAN and small world networks.

**CAN.** Content-addressable networks (CAN) was the first proposal using a multi-dimensional identifier space for structured overlay networks [Ratnasamy et al., 2001]. Identifiers are taken from  $m$ -dimensional torus  $[0, 1]^m$ . The identifiers are generated by applying a uniform hash function  $m$  times to the resources and peers. The  $m$ -dimensional torus is recursively split into *zones* till a peer is the sole occupant of a zone. This defines the resources it manages and the neighboring peers. The routing table maintains  $2m$  links to the peers in directly neighboring zones in the  $m$ -dimensional space. Query messages are forwarded to the neighbor with coordinates that are closest to the target in Euclidean metric. Thus, the routing cost is  $O(mn^{\frac{1}{m}})$ . A joining peer has to first locate the peer responsible for its identifier. It splits the zone of the peer currently responsible for its identifier into two and establishes the routing table. Thus, the cost of a peer join is  $O(mn^{\frac{1}{m}})$ . The approach of CAN can be extended by including long range links in exponentially increasing distances, similarly as in Chord and P-Grid, in order to achieve logarithmic routing cost [Xu and Zhang, 2002].

For structured overlay networks substantial research has been conducted to study the network performance and stability under realistic network conditions. We do not have the space to discuss those here in more detail but mention only some of the typical questions that have been addressed:

- *Maintenance.* An important performance characteristics is the behavior of a structured overlay network under churn, i.e., nodes joining and leaving the network [Gummadi et al., 2003]. Since structured overlay networks impose strict structural constraints on the network, it may become difficult to maintain those in case the churn exceeds a certain rate. The maintenance cost might become elevated, or it might even be impossible to maintain a consistent network structure.
- *Load balancing.* Even when using uniform hash functions to distribute resources among peers, it occurs that the resulting load at the peers is non-uniformly distributed [Rao et al., 2003]. Various techniques have been proposed to remedy this situation, including virtual peers, the power of two choices [Byers et al., 2003] and the use of multiple hash functions.
- *Replication.* Since peers are often unstable, resources might become inaccessible if they are not replicated within the peer-to-peer system. Typical replication schemes store a resource not only at the peer responsible for the resource but also at some neighboring peers in order to increase availability [Datta et al., 2003, Lv et al., 2002]. Also *erasure coding* has been considered for that purpose [Datta and Aberer, 2006, Weatherspoon and Kubiatowicz, 2002]. Erasure codes encode data by adding redundant data such that the original data can be recovered from any subset of the encoded data of which the size is above a threshold. Erasure codes have the advantage that they can achieve the same failure resilience as replication with less redundancy.

## 10 1. INTRODUCTION

Similarly, also the network links in the overlay network can be replicated in order to increase the failure resilience of the peer-to-peer overlay network itself.

- *Proximity-based routing.* Logical links in the overlay network might connect peers that are far apart in the underlay network, i.e., that are connected by long routes in the physical network. With proximity-based routing, the structure of the overlay network is aligned with the underlay network, such that neighboring links in the overlay network have also short connections in the physical network [Rowstron and Druschel, 2001].

An interesting observation is that most of the structured overlay networks follow a common underlying principle. They are realizations of *small-world graphs* that efficiently support greedy routing [Kleinberg, 2000]. Kleinberg showed that for a network of nodes that are connected in a regular grid by judiciously introducing long-range links the routing cost can be minimized for the greedy routing protocol. The standard structured overlay networks satisfy this necessary condition. Indeed, there exist a number of proposals for structured overlay network construction that directly apply the principles of small world graphs, either by explicit construction [Girdzijauskas et al., 2010, Manku et al., 2003] or as byproduct of query routing protocols [Clarke et al., 2001, Galuba and Aberer, 2007].

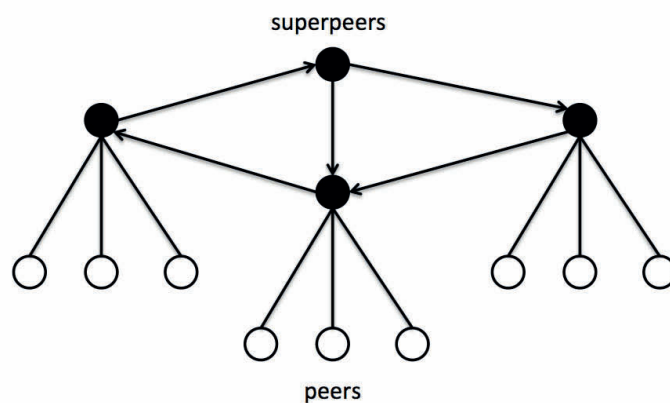
**Small world graphs.** In this model, peers are assumed to be connected in a  $m$ -dimensional regular grid which constitutes a network of short range links. Manhattan distance is used as distance metrics  $d$  on the grid. Search is performed by greedy routing. In order to accelerate greedy routing, long range links are added. In the case where per peer a constant number of long range links is added, it can be shown that for a grid with dimensionality  $m$  a long range link  $l(p_1, p_2)$  between peers  $p_1$  and  $p_2$  should be chosen with the following probability depending on their distance  $d(p_1, p_2)$ :

$$P[l(p_1, p_2)] \propto d(p_1, p_2)^{-m} .$$

Then the routing cost using greedy routing is  $O(\log^2 n)$ . By adding a logarithmic number of long range links, i.e., choosing the number of long range links proportional to  $\log n$ , the routing cost becomes  $O(\log n)$  [Girdzijauskas et al., 2005]. This model explains the behavior of most of the standard structured overlay networks mentioned above, such as Chord and P-Grid, as well as CAN when extended with long range links.

**Hierarchical overlay networks.** In practice, peers have widely diverse performance characteristics. Many nodes are unstable and remain in the peer-to-peer system only for short periods. Some nodes exhibit more server-like characteristics, have more available resources and stay in the system for extended periods. Based on this observation, most of the widely used practical peer-to-peer systems, like Gnutella, Emule, and Skype adopted a hierarchical approach for the peer-to-peer system architecture [Rasti et al., 2006]. More powerful peers, so called *superpeers*, take over a larger part of the system workload and behave towards less powerful, regular peers as servers. In a basic

superpeer architecture, a subset  $S \subset P$  of all peers take the role of superpeers. The superpeers connect in an unstructured or structured overlay network in order to route global search requests. Regular peers connect to one or more superpeers and forward search requests and updates to their superpeer. The superpeers typically maintain a global index of all resources available at the regular peers associated with them. In addition, the regular peers associated with a superpeer may form among themselves a local overlay network to process search requests. Only if local requests cannot be successfully resolved, they are forwarded via the superpeer to the global superpeer overlay network. Usually, superpeer networks are organized in a two-level hierarchy, but extensions using multiple levels of superpeer networks have been studied.



**Figure 1.4:** Illustration of a basic superpeer network. The superpeers are connected through an unstructured or structured overlay network.

**Gnutella ultrapeers.** In order to improve scalability of the Gnutella network, the ultrapeer concept has been introduced. An ultrapeer is a superpeer that is connected with other ultrapeers in a Gnutella unstructured overlay network. Different to the original Gnutella network, where each peer has an out-degree of typically 4, in an ultrapeer network, the out-degree is typically 30. Each ultrapeer maintains connections to up to 30 peers, and each peer attempts to connect to typically 3 ultrapeers. Thus, on average, 10% of the network consists of ultrapeers.

## 1.2 PEER-TO-PEER DATA MANAGEMENT

Peer-to-peer data management is a specific form of distributed data management, which considers large-scale distribution of data and a decentralized architectural approach. In a peer-to-peer data management system, at least one part of the system is implemented using a peer-to-peer architecture.

## 12 1. INTRODUCTION

Historically peer-to-peer data management has several roots. The first root is found in the extension of basic peer-to-peer systems, as described in the previous section, with richer search capabilities. Since resource location is a central problem for peer-to-peer systems, the problem of search has been always at the heart of the development of those systems. Enhancing search capabilities from simple key-based lookup to structured and similarity-based queries was a natural development, which was undertaken typically in the distributed systems and networking communities. The second root can be attributed to the adoption of the peer-to-peer architectural concept for distributed, federated and Web databases. Realizing that this architecture bears the potential of building more robust and scalable large-scale systems, the database and Semantic Web community started to apply it to generalize data integration and Web data management architectures. As a result, also novel overlay network structures derived from conventional database index structures were developed. Finally, a third root can be identified in the field of distributed information retrieval. Given that Web scale retrieval requires highly scalable solutions, it was a quite natural step to consider peer-to-peer architectural concepts for this problem.

Given the broad interest peer-to-peer data management has received in different communities and contexts, it is not surprising that the peer-to-peer approach has been applied to a wide range of data management problems. This includes standard, structured query processing, similarity search, data integration, continuous query processing, stream data processing, transaction processing, publish-subscribe and content distribution, just to mention the most important ones. In this book, we will cover a substantial fraction of these problems, but given the limited space, we have to omit some. We keep the focus on traditional ad-hoc data access, whereas different forms of active data and content distribution as well as support for transactions and workflows will not be covered.

Peer-to-peer systems have been developed to build large-scale distributed systems relying on decentralized control and self-organization. Applying this paradigm to data management implies a shift in some of the basic assumptions that are traditionally made in this field. This shift of assumptions also lays the ground to the development of novel techniques. We summarize and illustrate here some of the most important of these assumptions:

- *Large scale:* Distributed databases and information systems typically have been small to medium size involving up to a few hundred nodes. Peer-to-peer architectures consider systems with thousands to millions of nodes.
- *Decentralization:* Traditional distributed data management techniques typically include some form of centralized control, which may imply bottlenecks for a large scale system. Examples are root or coordinator nodes in distributed index structures, transaction coordinators in distributed transactions or global schemas in heterogeneous databases (see Chapter 4).
- *Communication cost:* In distributed data management, communication cost plays a less central role than in peer-to-peer systems, where it is considered the predominant cost. This aspect plays, for example, an important role in peer-to-peer retrieval where potentially long posting

lists need to be transferred. This is more critical in wide area networks than in local area networks, which are employed by today's search engine implementations (see Chapter 3).

- *Stability*: Failures are assumed to be a rare event in distributed data management, whereas they are the rule in a peer-to-peer system. This has implications for implementing operations such as hierarchical data aggregation, which might become highly unstable in a dynamic peer-to-peer setting (see Section 2.2.3).
- *Search paradigm*: The dominant query paradigm in peer-to-peer system is retrieval-oriented, even if search is performed on structured data (see Chapter 4). Due to the large scale, redundancy and volatility of a peer-to-peer system, as well as the type of data managed by peer-to-peer systems, including multimedia content and Web data, obtaining complete and consistent answers is much less of an issue than in well-structured and controlled database applications.

With the advent of cloud computing [Antonopoulos and Gillam, 2010], driven by large-scale Web applications, of course, also in the domain of cluster-based data management and analysis, similar shifts in the assumptions can be observed. Therefore, it does not surprise that concepts that have been incepted for the domain of peer-to-peer data management, such as overlay networks for data indexing, have also found their way into that domain today.

### 1.2.1 ARCHITECTURES

A peer-to-peer data management system introduces a *data management layer* on top of a peer-to-peer system. The peer-to-peer system provides services to the data management layer, allowing to realize different types of *physical data independence*. Physical data independence, in a nutshell, means that data access operations, such as value-based search or data scan, are specified at the level of applications logically. The underlying data management systems takes care of their optimized physical realization without the application being aware of the specific details of this. Peer-to-peer overlay networks support such physical data independence in two different flavors.

1. *Efficient data access*. Peer-to-peer overlay networks provide services that correspond to data access primitives that have been used in traditional data management systems for realizing physical data independence. This includes in particular indexing structures such as hash tables or B+-trees and table scans. These primitives support efficient data access under varying data distributions and storage schemes. Structured overlay networks provide a service comparable to classical database index structures, whereas both structured and unstructured overlay networks support by means of their broadcast capabilities a service comparable to classical database table scan.
2. *Isolation from network dynamics*. The peer-to-peer system handles the network dynamics, such as churn, changing network topology and performance variations for providing reliable resource access in an unstable system environment. It effectively makes these phenomena transparent

14 1. INTRODUCTION

to the data management layer. This property has been termed in the context of peer-to-peer data management as *network data independence* [Hellerstein, 2003].

We can distinguish two major classes of peer-to-peer data management systems, depending on how the data management layer is organized. In both cases, we assume that the data corresponds to some *data model*, which can be a structured or a multi-dimensional data model. We will call the data of such a data model the *data space*  $D$ .

In *homogeneous peer-to-peer data management systems*, the peers jointly manage one logical database  $R_D \subseteq D$  that is physically distributed in the network. The different peers provide equivalent logical access to the physically distributed data  $R_D$ . The data objects in  $R_D$  are considered as resources managed by the peer-to-peer system. The data is mapped from the *data space*  $D$  to the peer identifier space  $I$  by means of a mapping  $h_R : D \rightarrow I$ . The peer-to-peer overlay network of the peer-to-peer system provides the distributed access paths. This situation is depicted in Figure 1.5 on the left. As described in the previous section, the architecture of the peer-to-peer system consists of an overlay network on top of a communication network and peers are mapped to the identifier space  $I$  by means of a hash function  $h_P : P \rightarrow I$ . This type of system is the analog of traditional *distributed database system*. Homogeneous peer-to-peer data management systems will be the topic of Chapters 2 and 4.

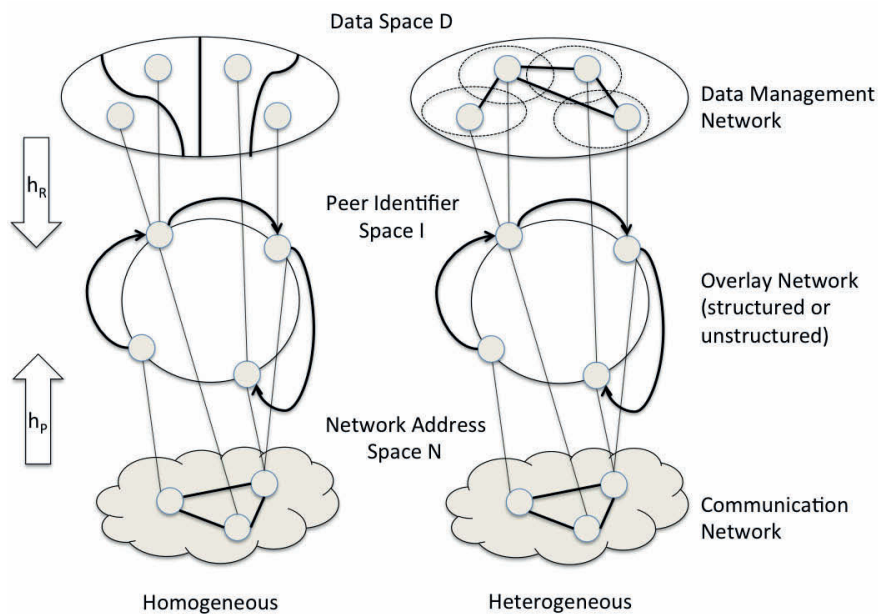


Figure 1.5: A generic architecture for peer-to-peer data management systems.

In a *heterogeneous peer-to-peer data management system*, the peers in the data management layer manage different logical databases, i.e., each peer  $p$  has a different *peer view*  $R_D(p)$  on the data space. As a result, the same operation executed at different peers may produce different results, in contrast to the case of homogeneous peer-to-peer management systems. We depict this architecture in Figure 1.5 on the right-hand side. The peer view of each peer, corresponding to the subspace of the data space that it can access, is indicated by the ellipses around the peers. In addition, peers are connected to other peers in the peer-to-peer network at the data management layer. The peers form a logical data management network that is embedded in the data space.

A link in the logical network may indicate different kinds of communication capabilities among peers. In the simplest case, it corresponds to the ability to query and update data using a common protocol and data format. A link may carry information about the semantic similarity of the data managed by the peers. In more complex situations, a link may correspond to the capability of performing complex data processing, in particular mapping among different formats and types of data.

For the data management network, the data space  $D$  can play the analogous role as the identifier space  $I$  plays for a structured overlay network. The data space may be equipped with a distance or similarity metrics, similarly as for identifier spaces in structured overlay networks, and greedy routing is used as search strategy, but this needs not always to be the case.

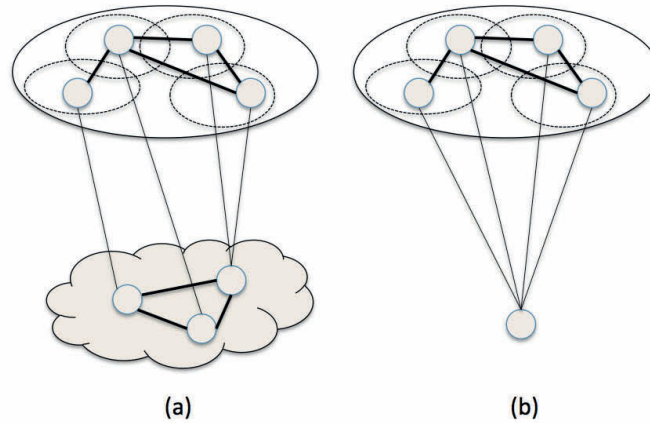
In a heterogeneous peer-to-peer data management system, the relationship between the data management network and the overlay network, is analogous to the relationship among the overlay network and the physical network, for the peer-to-peer system. Different peers may be mapped to the same node in the overlay network and logical links among peers in the data management network may translate to multiple hops in the overlay network. Heterogeneous peer-to-peer data management systems are the analog of *federated databases* in traditional data management. Usually, only few heterogeneous peer-to-peer data management systems follow this architecture in its full generality. Frequently, different special cases of it are considered. These are shown in Figure 1.6.

Figure 1.6(a) shows the case where the structure of the data management network and of the overlay network coincide. This situation typically occurs when the identifier space of an overlay network coincides with the data space of an application. Figure 1.6(b) shows the situation where the peer-to-peer interactions take place only at a logical level, whereas physically the system is implemented in a single, centralized network node. Heterogeneous peer-to-peer data management systems will be the topic of Chapters 3 and 5.

### 1.2.2 DESIGN DIMENSIONS

For both the overlay network layer and the data management layer, a wide range of design choices is available. These design choices depend on the application requirements as well as on the performance objectives. We summarize in the following key design dimensions.

For the implementation of the overlay network structure in the peer-to-peer data management architecture, we can distinguish three approaches.



**Figure 1.6:** Special cases of the generic architecture for heterogeneous peer-to-peer data management systems.

1. *Use of existing overlay networks as black box.* In this approach, existing overlay networks are used in their original design. The advantage is that existing implementations and deployments can be used in their unmodified form. The disadvantage is that their performance characteristics or function might not be adapted to the needs of data management applications.
2. *Adaptation of overlay networks to data management requirements.* For the above reason in many approaches, existing overlay network designs are modified to better suit the needs of data management applications, like support for more complex query types with the resulting load balancing issues (see Chapter 2). A disadvantage of this approach can be that such modifications might make the design more complex or existing overlay network approaches do not permit to achieve specific design goals.
3. *Design of new overlay networks inspired by database indexing structures.* Thus, as a final alternative, novel overlay networks can be developed that are typically decentralized versions of classical database indexing structures, such as B+-Trees. Such overlay networks may indeed exhibit desirable properties from a data management perspective, but they may also sacrifice some principles of peer-to-peer systems, like requiring central control, and may suffer from high implementation complexity.

At the data management layer, the choice of the data model is a key distinguishing factor for different approaches to peer-to-peer data management. The data model is determined by the application needs and different data models may imply significantly different algorithmic problems that have to be tackled. We find for peer-to-peer data management the following typical data models.

- *Low-dimensional multi-attribute data*: This model is typically used for resource description and search. Data in this model can be understood as single relational table with multiple attributes. Typical operations are key-based lookups, value-based searches, range searches, conjunctive queries on multiple attributes and aggregation queries.
- *Structured databases with schema*: This is the standard data model used in traditional database systems, with the relational data model and XML as the main representatives. It can be understood as a generalization of the previous model. Data access is supported by full-fledged query languages, which implies in particular the capability to express join queries.
- *Semi-structured data*: This model is used in document management and Web data management, with RDF and XML as the main representatives. In this model, data is self-describing and tree- or graph-structured. A typical additional processing requirement as compared to the previous models results from the need to support path or graph pattern queries.
- *High-dimensional feature data*: This model is typically used for content retrieval. Data items are points in high dimensional metric spaces. Typical operations in this model are similarity searches, such as nearest neighbor or top-k queries.

Each of these models has been considered for homogeneous peer-to-peer data management systems. For heterogeneous peer-to-peer data management systems, we find two main variants.

- *Structured data with peer mappings*: In this approach, peers have different views on the data space by using structurally different representation for the same type of data. Links among peers correspond to the capability of mapping among different structural representations. This type of peer-to-peer data management system is called *peer-to-peer data integration systems*.
- *Multi-dimensional data with peer similarity*: In this approach, peers have different views on the data space by specializing on and covering certain subregions. These subregions correspond to interest or expertise profiles of peers. Links among peers correspond to forward queries to other peers that have similar profiles. This type of peer-to-peer data management system is called *semantic overlay network*.

### 1.2.3 OVERVIEW OF THE LECTURE

The forthcoming four chapters of the lecture will introduced four different types of peer-to-peer data management systems that are characterized by the type of data they manage and the search capabilities they support.

In Chapter 2, we discuss *structured peer-to-peer data management systems* which support structured query capabilities for standard data models. This type of peer-to-peer data management systems falls into the class of homogeneous peer-to-peer data management systems. The data these systems manage is either relational or RDF structured data. A key problem is the partitioning scheme that

is used to distribute the data over the peers. Operations that are supported in this type of systems range from simple multi-attribute queries to complex join and aggregation queries.

The following Chapter 3 is dedicated to *peer-to-peer data integration systems*. These support querying of databases that are using heterogeneous database schemas without requiring a common global schema. They belong to the class of heterogeneous peer-to-peer data management systems. The data in these systems is typically relational, XML or RDF. The key problem for these system is the establishment and use of mappings to overcome heterogeneity among different schemas. These systems support usually full-fledged structured query languages.

Next, in Chapter 4, we focus on *peer-to-peer document retrieval systems* that enable document search based both on the textual content and the document structure, e.g., represented as XML. This type of system is another frequently encountered example of homogeneous peer-to-peer data management systems. A key problem for this type of system is to efficiently index document features, both text and structural features. Since these can be highly non-uniformly distributed, this poses significant challenges for load balancing and keeping communication cost acceptable when searching based on those features.

Finally, in Chapter 5, we introduce *semantic overlay networks*, which in a sense generalize the previous types to systems that manage data from both hierarchically organized and multi-dimensional semantic spaces that are equipped with a similarity measure to model semantic proximity. They are another example of heterogeneous peer-to-peer data management systems. A key question for this type of systems is to find mappings from the data space to the identifier space of peer-to-peer overlay networks. Since, in the data space, proximity is used to represent semantic similarity, these mappings should be proximity-preserving. This allows the system to efficiently handle similarity-based queries, as typically needed for content retrieval and information search.

We conclude the lecture in Chapter 6, by highlighting topics that we have not addressed in this lecture and that either belong to the realm of peer-to-peer data management or are closely related to it.