

Chapter 2

Peer-to-Peer Systems

Abstract In this chapter, a basic overview is given of P2P systems, architectures, and search strategies in P2P systems. More specific concepts that are outlined include the differences of structured and unstructured P2P systems, categories of P2P systems based on the centralization degree, basic search mechanisms for unstructured P2P systems, as well as details on P2P data management systems.

2.1 General Characteristics

P2P systems refer to distributed computer architectures that are designed for sharing resources, by direct exchange, rather than requiring a central coordinating server. In P2P systems, the interconnected computers, called *peers*, are organized in a network in a distributed and self-organizing way and share resources, while respecting the autonomy of peers. The main characteristic of a P2P system is the ability to adapt to peer failures (fault-tolerance) and accommodate a large number of participating peers (scalability), while keeping the performance of the network at an acceptable level and maintaining the peer connectivity.

Generally, all peers are equivalent in terms of tasks and functionality they perform, and act both as clients and servers at the same time. Peers participate in the system in a collaborative manner, by performing tasks such as connecting to other peers, routing messages to other peers and searching for content. With respect to the peer autonomy, each peer maintains and controls its own content or resources.

Even though different kinds of resources may be shared over a P2P network, such as CPU or storage, P2P systems emerge as a powerful model for organizing and searching of huge amounts of data distributed over independent sources. Applications such as file-sharing, distributed database and digital libraries gain from such an architecture. Therefore, in this book, we focus on P2P systems that share content, one of the most prominent applications areas of P2P technology.

Each peer has a collection of files or data to share. Two peers that maintain an open connection between them are called *neighbors*. The number of neighbors of a

node defines its *outdegree* or *degree*. Any peer can pose a query, in order to retrieve interesting content; this peer is called *querying peer*. When a peer receives a query, first the query is evaluated against its local data collection and thereafter, if necessary other peers are contacted through its neighbors. Query messages are forwarded only between open connections, i.e., neighboring peers. If a message has to be exchanged between two non-neighboring peers, more than one message is required. The cost of this message is considered that it is proportional to the length of the path, also called number of *hops*. Usually query results are forwarded back to the querying peer through the reverse path of the query message. Alternatively, some approaches allow the establishment of a temporary connection, in order to transfer the query results directly to the querying peer.

In the rest of this chapter, we will describe in more detail different architectures for P2P systems, how to search for data in such systems, and finally we give an overview of some P2P-based data management systems.

2.2 P2P Architectures

P2P systems can be classified into two categories, based on the way the content is located in the network, with respect to the network topology: *unstructured* and *structured*. In unstructured P2P, each peer maintains a limited number of connections (also called links) to other neighboring peers in the network. Searching in an unstructured P2P environment usually leads to either flooding queries in the network using a time-to-live (TTL) or query forwarding based on constructed routing indices that give a direction for the search. Examples of such unstructured P2P networks include Gnutella and Freenet [39]. In structured P2P systems, a hash function is used in order to couple keys with objects. Then a distributed hash table (DHT) is used to route key-based queries efficiently to peers that hold the relevant objects. In this way, object access is guaranteed within a bounded number of hops. Examples of popular structured P2P systems are Chord [121], CAN [106], Pastry [112], Tapestry [146].

2.2.1 Structured P2P Systems

Structured P2P systems assume a relation between the peer content and the network topology. The topology of the network is strongly controlled and data is stored or indexed by specific peers. Therefore, there is a mapping between the data objects and the location, i.e., the peer that stores or indexes it, so that queries can be routed easily to peers that store relevant data objects. Usually the mapping is realized as a distributed hash table (DHT). We shortly describe two representative structured P2P networks, namely CHORD [121] and CAN [106].

CHORD [121] is a P2P protocol that uses a distributed hash table for efficiently locating the peer that stores the data item corresponding to a given one-dimensional key value. CHORD uses *consistent hashing* to map uniformly the domain of search keys into the Chord domain of keys. More precisely, each peer is identified by an m -bit key value generated by the hash function applied on the peer's IP. Then, the peers are ordered in an *identifier circle* based on their key value. Each object with a key value k is assigned to the first peer that has a key value equal to or larger than k . In order to maintain efficient communication between peers, every peer stores the addresses of its predecessor and successor on the identifier circle. In addition, each peer maintains a routing table called the *finger table* with addresses of up to m other nodes. With high probability, the peer responsible for a given key value is located via $O(\log(N))$ number of messages to other peers.

CAN [106] divides dynamically the d -dimensional data space into partitions, so that each peer is assigned to a data space partition. Each peer connects to the peers that store neighboring data space partitions, according to some dimension. Each time a peer joins the network, it randomly chooses a point in the d -dimensional space and contacts the peer responsible for the partition in which that data point belongs. Then this peer, splits the data space in two parts, based on some dimension and then assigns the one part to the new peer. Finally, the neighboring peers are contacted and updated. A point query based on the CAN architecture is performed in $O(N^{1/d})$ messages.

2.2.2 Unstructured P2P Systems

In contrast to structured P2P networks, in unstructured P2P networks peers connect to other peers in a random way and there is no relation of the placement of the data objects with the network topology. Therefore, peers have no or limited information about data objects stored by other peers, thus searching in unstructured P2P networks may lead to query all neighbors for data items that match the query (see Section 2.3).

Unstructured P2P networks are easy to implement and impose low maintenance cost, however query routing is not scalable. As the number of participant peers increases, the number of exchanged messages increases too. On the other hand, structured P2P networks maintain information about the data objects available through their neighbors. Therefore, query processing algorithms with cost bounds have been proposed. Nevertheless, the maintenance cost is high, especially for large P2P networks and high rates of peer joins or departures (also known as *churn*).

2.2.3 P2P Network Centralization Degree

P2P overlay networks were initially defined to be totally decentralized, however for practical reasons, systems with different degrees of centralization have been developed. Therefore, except of purely decentralized architectures, also hybrid systems were proposed.

In a purely distributed P2P system, all nodes in the network perform exactly the same tasks, acting both as clients and servers, and there is no central coordination of their activities. The major shortcomings of purely distributed systems is scalability issues and the poor performance during query processing.

In the hybrid centralized indexing systems, there is a central server facilitating the interaction between peers and a centralized index is build at this specialized peer. Napster, for example, uses a centralized index, that is built in cooperation with all the participating peers. The centralized index keeps information about the data stored at each peer, together with the peer identifier. Therefore, centralized indices are efficient during query processing; a single message is required to determine which peer stores relevant information. Notice that the actual sharing of information between peers is established by communication between the peers, without interaction with the central server. Despite the efficiency during query processing, centralized indices have a major drawback, namely they constitute a "single point of failure". Moreover, the centralized index may become a bottleneck for the system, especially in the case of a large P2P network.

Hybrid decentralized indexing systems, namely super-peer infrastructures [141], harness the merits of both centralized and purely distributed architectures. Super-peer networks tackle the scaling and "single-point-of-failure" problems of centralized approaches, while exploiting the advantages of the completely distributed approach, where each peer builds and maintains an index over its own files. These systems are similar to purely decentralized systems, but some of the peers have a more important role, and are responsible to maintain the information available at their associated peers and facilitate the interaction between peers. If only the super-peers are considered, they act as a purely decentralized P2P system.

2.3 Search in Unstructured P2P Systems

Different approaches have been proposed for unstructured P2P systems, in order to find interesting content for the users.

2.3.1 Flooding-Based Search

The most naive approach that does not use any index is *flooding* the network, until interesting content is retrieved. Flooding is performed by each peer forwarding the

query to all neighbors, except from the one that the query was received. It is obvious, that flooding-based search does not constitute a truly scalable solution, as the query-induced traffic practically saturates the P2P system. Therefore, several variations have been proposed, in order to reduce the cost of flooding.

An alternative to pure flooding is to attach a *time-to-live* (TTL) that specifies after how many hops the query should be stopped, which leads to the problem of a limited horizon. Any peer receiving a query, first decrements the TTL and only if the TTL is greater than zero it forwards the query to its neighbors. Flooding was used by Gnutella. Although this approach is simple and there is no overhead for maintaining routing information, it has the disadvantage that it may lead to an enormous cost during query processing, especially in the case of large P2P networks. In addition, flooding is sensitive to the connectivity degree of each peer, since a small connectivity degree leads to long routing paths, while in a dense network topology each peer receives the query from multiple paths, resulting in a large number of exchanged messages and waste of bandwidth.

Another variant is to perform a *random walk* on the network [60]. Using this strategy, the query is forwarded to one or some random neighbors instead of forwarding to all neighboring peers.

2.3.2 Routing Indices

Routing indices [42] were proposed to alleviate the shortcomings of the aforementioned search methods. Routing indices are implemented as multiple small indices that are stored distributed at every peer, that differs to a costly and vulnerable centralized index. Routing indices store the direction that should be followed, in order to find the requested data, rather than the corresponding data itself. In more detail, each peer maintains an index that describes the information that is available through each neighboring peer. For example, in the case of documents, a centralized index would store the terms with the peers' IPs that store documents containing the terms, while a routing index would maintain only information about how many documents containing each term we could find by following a path starting at each neighboring peer. Therefore, each peer is able to forward a query to the neighbors that are more likely to have results, instead of choosing some neighbors randomly or flooding the network by forwarding the query to all neighbors. The routing indices are usually constructed assuming an acyclic network topology, while different approaches are proposed for handling cycles.

2.3.3 Semantic Overlay Networks

In application areas where data is non-uniformly distributed over the peers, and it is possible to discover peers with similar content, then it is possible to form the-

matically focused peer groups and use this for query routing. This technique, called *semantic overlay networks* [53], makes it possible to route queries to only those peer groups that are relevant to the query, thus significantly reducing the number of peers that has to be contacted during a query.

2.4 Super-Peer Systems

The choice of the underlying network architecture plays an important role in the performance of a P2P system. However, no single architecture is suitable for all types of possible P2P applications. Super-peer infrastructures [141] harness the merits of both centralized and distributed architectures, as they tackle both the scaling limitations and the "single-point-of-failure" problem of centralized approaches, while exploiting the advantages of the completely distributed approach, where each peer builds an autonomous index over its own files. Nowadays, super-peer architectures have been established in most of the existing P2P file-sharing applications (eMule, KaZaA).

Moreover, an important performance issue in P2P networks is uneven load distribution that can lead certain points of the network to become a bottleneck, caused by the limited capabilities of some peers. Super-peer networks take advantage of the heterogeneity of peer capabilities (e.g., bandwidth, processing power), which recent studies have shown to be enormous [117], by assigning greater responsibility to those that are more capable of handling it. In addition, in several applications, cooperative computers in a P2P network have different roles by nature, similar to the case of file-sharing, where some machines are registered as dedicated servers to the system, while others are plain personal computers that mostly request information. Furthermore, in order to respect peers autonomy, any approach should not rely on arbitrary data movement, hence each peer joining the network should autonomously store its own data. Therefore, a super-peer architecture appears particularly suitable for applications that require efficient performance for advanced query operators, hence we model our distributed system as a super-peer network.

Numerous interesting applications that require efficient query processing can be deployed over such a super-peer architecture. The overall objective is for a set of cooperative computers to collectively provide enhanced processing facilities, aiming at overcoming the limitations of centralized settings, for example extremely high computational and storage load. Examples of challenging applications that can be realized over the proposed framework include distributed image retrieval, document retrieval in digital libraries, P2P information sharing, data integration from distributed sources, as well as distributed scientific databases.

2.5 P2P-Based Data Management Systems

During the last decade a number of P2P-based data management systems have been realized. Some provides just basic storage capabilities, while others also support advanced query capabilities.

OceanStore [88] is one of the storage systems without query capabilities. It provides an infrastructure for permanent storage and replication of objects, but no query system. Objects are accessed based only on their globally unique ID, and this ID has to be known in order to retrieve or update the object.

PIER [77] is a middleware query engine built on top of a DHT. PIER does not permanently store its data. Data sources publish their data in the DHT and update them regularly, and data that are not refreshed are removed. Typically, a PIER network will contain only object metadata (e.g., filenames, sizes, tags) and a reference to the original data object. Clients will query the network to get the references to the objects of interest and retrieve the objects separately.

Among the systems that provide a full DBMS, with both query processing and storage, are Hyperion [110], Orchestra [126], Piazza [66]. All these systems allow each site to have its own schema, and use schema mediation techniques to allow cross-site querying. PeerDB [99] also falls into this category of systems with heterogeneous schemas, but the approach to schema mediation is different. Instead of relying on schema mediators, information retrieval techniques are used to find matching relations.

A slightly different approach is DASCOSA-DB [67], which does not use schema mediation. While the systems mentioned above are meant to connect existing databases and provide a common query interface, DASCOSA-DB is a distributed database system with a high degree of site autonomy, but which still behaves as one system, not many different systems with a common interface.

Other systems based on a common schema include APPA [1]. APPA provides a multilayered solution on top of a structured or super-peer P2P network, where the bottom layer is a simple key/value-store and the top level provides advanced services such as schema management, replication and query processing.

AmbientDB [20] is a system designed to provide full relational database functionality for stand-alone operation in autonomous devices that may be mobile and disconnected for long periods of time, while enabling them to cooperate in an ad hoc way with (many) other AmbientDB devices. A DHT is used both as a means for connection peers in a resilient way as well as supporting indexing of data. AmbientDB is a system for mobile devices, which have low computational power and may frequently be disconnected from the network, compared to for example DASCOSA-DB that is designed for sites that have the computational power necessary to do query processing and more stable network connections.

Peer-to-Peer Query Processing over Multidimensional
Data

Vlachou, A.; Doulkeridis, C.; Norvag, K.; Kotidis, Y.

2012, XII, 84 p. 18 illus., Softcover

ISBN: 978-1-4614-2109-2