Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Approximate Information Filtering in Structured Peer-to-Peer Networks

Christian Zimmer

Max-Planck Institute for Informatics

Saarbrücken
2008

max planck institut
informatik

Promotionskolloquium

Tag des Promotionskolloquiums	30. Oktober 2008
Ort des Promotionskolloquiums	Saarbrücken
Dekan der Naturwissenschaftlich-Technischen Fakultät I	Prof. Dr. Joachim Weickert *Universität des Saarlandes*

Prüfungsausschuss

Vorsitzender	Prof. Dr. Jens Dittrich *Universität des Saarlandes*
Gutachter	Prof. Dr.-Ing. Gerhard Weikum *Max-Planck Institute for Informatics*
Gutachter	Prof. Dr. Manolis Koubarakis *National and Kapodistrian University of Athens*
Gutachter	Dr. Christos Tryfonopoulos *Max-Planck Institute for Informatics*
Akademischer Beisitzer	Dr.-Ing Ralf Schenkel *Universität des Saarlandes*

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, den 14. Juli 2008

Christian Zimmer
(Unterschrift)

Contents

List of Figures

List of Tables

Abstract

Today's content providers are naturally distributed and produce large amounts of information every day, making peer-to-peer data management a promising approach offering scalability, adaptivity to dynamics, and failure resilience. In such systems, subscribing with a continuous query is of equal importance as one-time querying since it allows the user to cope with the high rate of information production and avoid the cognitive overload of repeated searches. In the information filtering setting users specify continuous queries, thus subscribing to newly appearing documents satisfying the query conditions.

Contrary to existing approaches providing exact information filtering functionality, this doctoral thesis introduces the concept of approximate information filtering, where users subscribe to only a few selected sources most likely to satisfy their information demand. This way, efficiency and scalability are enhanced by trading a small reduction in recall for lower message traffic.

This thesis contains the following contributions: (i) the first architecture to support approximate information filtering in structured peer-to-peer networks, (ii) novel strategies to select the most appropriate publishers by taking into account correlations among keywords, (iii) a prototype implementation for approximate information retrieval and filtering, and (iv) a digital library use case to demonstrate the integration of retrieval and filtering in a unified system.

Kurzfassung

Heutige Content-Anbieter sind verteilt und produzieren riesige Mengen an Daten jeden Tag. Daher wird die Datenhaltung in Peer-to-Peer Netzen zu einem vielversprechenden Ansatz, der Skalierbarkeit, Anpassbarkeit an Dynamik und Ausfallsicherheit bietet. Für solche Systeme besitzt das Abonnieren mit Daueranfragen die gleiche Wichtigkeit wie einmalige Anfragen, da dies dem Nutzer erlaubt, mit der hohen Datenrate umzugehen und gleichzeitig die Überlastung durch erneutes Suchen verhindert. Im Information Filtering Szenario legen Nutzer Daueranfragen fest und abonnieren dadurch neue Dokumente, die die Anfrage erfüllen.

Im Gegensatz zu vorhandenen Ansätzen für exaktes Information Filtering führt diese Doktorarbeit das Konzept von approximativem Information Filtering ein. Ein Nutzer abonniert nur wenige ausgewählte Quellen, die am ehesten die Anfrage erfüllen werden. Effizienz und Skalierbarkeit werden verbessert, indem Recall gegen einen geringeren Nachrichtenverkehr eingetauscht wird.

Diese Arbeit beinhaltet folgende Beiträge: (i) die erste Architektur für approximatives Information Filtering in strukturierten Peer-to-Peer Netzen, (ii) Strategien zur Wahl der besten Anbieter unter Berücksichtigung von Schlüsselwörter-Korrelationen, (iii) ein Prototyp, der approximatives Information Retrieval und Filtering realisiert und (iv) ein Anwendungsfall für Digitale Bibliotheken, der beide Funktionalitäten in einem vereinten System aufzeigt.

Summary

The present doctoral thesis with the title *Approximate Information Filtering in Structured Peer-to-Peer Networks* introduces a novel approach to manage continuous queries in an information filtering environment over a network of autonomous publishers and subscribers. This thesis combines research from two active research areas in computer science: peer-to-peer networks, and information retrieval and filtering. The work presented here is the first approach to study information filtering in a dynamic environment by relying on the novel concept of approximate information filtering.

Information filtering, also referred to as publish/subscribe or continuous querying or information push, is equally important to information retrieval (or one-time querying), since users are able to subscribe to information sources and be notified when new events matching their information demand occur. Information filtering and information retrieval are often referred as two sides of the same coin, since many of the underlying issues and goals are similar; in both cases a document needs to be matched against an information demand. Despite this duality between retrieval and filtering, the design issues, the techniques and algorithms devised to increase filtering efficiency differ significantly from those utilized by retrieval.

In the last years, a lot of research efforts have been concentrated on providing distributed information management. With the proliferation of peer-to-peer systems, mainly due to file-sharing applications such as Gnutella or BitTorrent, peer-to-peer information management has gained increasing attention. The main advantage of such systems is the ability to handle huge amounts of data in a decentralized and self-organizing manner offering high potential benefit for information systems powerful regarding scalability, efficiency, and resilience to failures and dynamics. Contrary to server-oriented architectures, peer-to-peer systems avoid single-points-of-failure, and such an information system can potentially benefit from the intellectual input (e.g., click streams or query logs) of a large user community participating in the data-sharing network.

This doctoral thesis builds upon research in peer-to-peer and information filtering to introduce a novel architecture and the first approach for approximate information filtering in distributed networks. Most approaches on information filtering taken so far have the underlying hypothesis of potentially delivering notifications from every information producer to subscribers. This exact information filtering model imposes a cognitive overload on the user in the case of applications like blog or news filtering, and creates an efficiency and scalability bottleneck. Contrary to this, the novel approximate information filtering approach ranks sources, and delivers matches only from the best ones, by utilizing novel publisher selection strategies. Thus, the continuous query is replicated to the best information sources and only published documents from these sources are forwarded to the subscriber. This approximate information filtering relaxes the assumption, which holds in most filtering systems, of potentially delivering notifications from every producer and amplifies scalability.

The architecture presented in this thesis provides the tools and protocols to identify the most appropriate publishers for a given continuous query by exploiting metadata stored in a conceptually global, but physically distributed directory. To select the most appropriate publishers to subscribe to, a subscriber computes scores that reflect the past publishing behavior of information sources and utilizes them to predict their future publishing behavior. These scores are based on a combination of resource selection and behavior prediction to deal with the dynamics of publishing. Behavior prediction uses time-series analysis with double exponential smoothing techniques to predict future publishing behavior, and adapt faster to changes in it. In addition, correlations among keywords in multi-keyword continuous queries can be exploited to further improve publisher selection. For scalability reasons, the metadata stored in the directory has publisher and not document granularity, thus capturing the best publishers for certain keywords. Building on the main architecture, the thesis introduces two new algorithms to exploit correlations among keywords to improve publisher selection. The first algorithm uses single-keyword synopses stored in the directory to estimate the publishing behavior of information sources for sets of keywords, while the second algorithm enhances the distributed directory to explicitly maintain statistical metadata about selectively chosen sets. Existing, self-limited approaches for two-keyword queries are extended to the case of multi-keyword continuous queries for an arbitrary number of keywords. Beyond that, the thesis presents algorithms to approximate multi-keyword statistics by combining the metadata statistics of arbitrary subsets. By utilizing the proposed techniques approximate filtering achieves higher scalability than exact filtering by trading faster response times and lower message traffic for a moderate loss in recall.

Finally, the thesis presents a digital library use case that unifies approximate retrieval and filtering functionality under a common architecture, and a prototype implementation that showcases the applicability of the proposals. Comparative experiments conducted on several real-world data sets quantify the efficiency and effectiveness of the methods proposed in this thesis.

Zusammenfassung

Die vorliegende Doktorarbeit mit dem Titel *Approximate Information Filtering in Structured Peer-to-Peer Networks* stellt einen neuen Ansatz zur Verarbeitung von Daueranfragen in einer Information Filtering Umgebung mit einem Netzwerk aus unabhängigen Anbietern und Beziehern vor. Die Arbeit verknüpft aktuelle Forschung aus zwei aktiven Forschungsbereichen der Informatik: Peer-to-Peer Netzwerke und Information Retrieval und Filtering. Die vorgestellte Arbeit ist der erste Ansatz für Information Filtering in einer dynamischen Umgebung und beruht auf dem neuen Konzept von approximativem Information Filtering.

Information Filtering – auch bezeichnet als Publish/Subscribe, dauerhaftes Anfragen oder Information Push – ist genauso wichtig wie Information Retrieval (oder einmaliges Anfragen). Der Grund liegt darin, dass Benutzer in der Lage sind, Informationsquellen zu abonnieren, so dass sie über neue Ereignisse benachrichtigt werden, die ihrem Informationsbedürfnis entsprechen. Information Filtering und Information Retrieval werden oft als zwei Seiten einer Medaille bezeichnet. Obwohl viele der Probleme und Ziele von Retrieval und Filtering ähnlich sind – schließlich werden in beiden Fällen Dokumente mit einem Informationsbedürfnis verglichen –, so unterschieden sich Designfragen, sowie Techniken und Algorithmen zur Verbesserung der Effizienz beim Filtering doch deutlich.

In den letzten Jahren konzentrierte sich viel Forschungsarbeit auf die verteilte Datenhaltung. Mit dem Aufstieg von Peer-to-Peer Systemen, hauptsächlich durch File-Sharing Anwendungen wie Gnutella oder BitTorrent, erfuhr auch die Peer-to-Peer Datenhaltung eine verstärkte Aufmerksamkeit. Der wichtigste Vorteil solcher Systeme liegt in der Fähigkeit, große Datenmengen auf eine verteilte und selbstorganisierende Art und Weise zu handhaben. Die Charakteristiken von Peer-to-Peer bieten großes Potential für Informationssysteme in Bezug auf Skalierbarkeit, Effizienz und Widerstandsfähigkeit gegenüber Fehlern und Dynamik. Im Gegensatz zu Server-orientierten Architekturen vermeiden Peer-to-Peer Systeme einen einzelnen Fehlerpunkt. Darüber hinaus kann ein solches Informationssystem möglicherweise vom intellektuellen Input (in Form von Click-Streams oder Anfrage-Logs) einer großen Benutzergemeinschaft profitieren.

Diese Doktorarbeit basiert auf Forschung in den Bereichen Peer-to-Peer Netzwerke und Information Filtering, um eine neuartige Architektur und den ersten Ansatz für approximatives Information Filtering in verteilten Netzwerken einzuführen. Die meisten bisherigen Ansätze besitzen die zugrunde liegende Annahme, Benachrichtigungen aller Informationsquellen zu liefern. Dieses exakte Information Filtering Model führt zu einer erkennbaren Überlastung des Nutzers für Anwendungen wie Filtering von Blog- oder Nachrichtenbeiträgen. Außerdem wird ein Effizienz- und Skalierbarkeitsproblem erzeugt. Der ganz neue Ansatz des approximativen Information Filtering hingegen ordnet Informationsquellen mit neuen Strategien zur Auswahl von Anbietern. Nur Treffer der besten Anbieter werden geliefert, indem eine Daueranfrage auch nur an die besten Informationsquellen geschickt wird, welche dann neue Dokumente an den Abonnenten weiterleiten. Approximatives Information Filtering schwächt die Annahme der meisten Filtering Systeme, Benachrichtigungen von allen möglichen Anbietern zu liefern, und verstärkt dadurch Skalierbarkeit.

Die in dieser Arbeit präsentierte Architektur bietet die Werkzeuge und Protokolle, um die am meisten geeigneten Anbieter für eine Daueranfrage zu ermitteln, indem Metadaten genutzt werden, die in einem konzeptionell globalen, aber physisch verteilten Verzeichnis gespeichert werden. Ein Abonnent berechnet zur Wahl der besten Anbieter Bewertungszahlen, die das bisherige Verhalten beinhalten und dieses nutzen, um zukünftiges Verhalten vorauszusagen. Diese Bewertungszahlen basieren auf einer Kombination aus Resource Selection und Behavior Prediction, um die gegebene Dynamik zu berücksichtigen. Behavior Prediction benutzt Zeitreihenanalyse mit Double Exponential Smoothing Techniken zur Vorhersage zukünftigen Verhaltens mit schnellen Veränderungen. Zusätzlich können Korrelationen innerhalb der Schlüsselwörter einer Anfrage ausgenutzt werden, um die Auswahl der Anbieter weiter zu verbessern. Aus Skalierbarkeitsgründen besitzen die Metadaten, die im Verzeichnis gespeichert werden, Anbieter-Granularität und keine Dokumenten-Granularität. Dadurch werden die besten Anbieter für einzelne Schlüsselwörter festgestellt. Auf der Hauptarchitektur aufbauend bietet diese Arbeit zwei Algorithmen, um Korrelationen unter Schlüsselwörter zur Verbesserung der Auswahl der Anbieter auszunutzen. Der erste Algorithmus benutzt individuelle Schlüsselwörter Synopsen aus dem Verzeichnis, um die Auswahl zu verbessern. Der zweite Algorithmus erweitert das verteilte Verzeichnis um Metadaten von explizit ausgewählte Mengen aus mehreren Schlüsselwörtern. Bereits existierende Ansätze, die sich auf Mengen aus zwei Schlüsselwörtern beschränken, werden auf beliebig viele Schlüsselwörter erweitert. Darüber hinaus zeigt diese Arbeit einen Algorithmus auf, welcher Statistiken für Mengen aus mehreren Schlüsselwörtern durch die Kombination von beliebigen Teilmengenstatistiken approximiert. Durch Anwendung der vorgeschlagenen Techniken erhält approximatives Information Filtering eine höhere Skalierbarkeit. Für einen akzeptablen Verlust an Recall erhält man schnellere Antwortzeiten und einen geringeren Nachrichtenverkehr.

Schließlich präsentiert diese Doktorarbeit einen Anwendungsfall für Digitale Bibliotheken, welcher approximative Retrieval und Filtering Funktionalität in einer vereinten Architektur unterstützt, und eine prototypische Implementierung, die die Anwendbarkeit von approximativem Information Filtering herausstellt. Vergleichende Experimente auf unterschiedlichen echten Datenmengen messen die Effizienz und Effektivität der Methoden dieser Arbeit.

Chapter 1

Introduction

This introductory chapter motivates the doctoral thesis in Section 1.1 by introducing the problem of P2P IF. Section 1.2 highlights the main solution and presents the major contributions that will be investigated in the following chapters, while Section 1.3 presents the thesis outline.

1.1 Motivation

The peer-to-peer (P2P) paradigm has been receiving increasing attention in recent years, mainly in the context of file-sharing applications (*Gnutella*, *BitTorrent*) or other Internet applications such as IP telephony (e.g., *Skype*). In the meantime, the P2P paradigm is moving towards *distributed data management systems* such as *information retrieval* (IR) or *information filtering* (IF). The main advantage of P2P is its ability to handle huge amounts of data in a decentralized and self-organizing manner. The characteristics of P2P offer high potential benefit for information systems powerful regarding scalability, efficiency, and resilience to failures and dynamics. Beyond that, such an information system can potentially benefit from the intellectual input of a large user community participating in the data sharing network. Finally, P2P information systems can also bring forward pluralism in informing users about Internet content, which is crucial in order to guard against information-resource monopolies and the biased visibility of content from economically powerful sources. Information censorship is often mentioned in this context.

Information filtering, also referred to as *publish/subscribe* or *continuous querying* or *information push*, can be seen as equally important to information retrieval (or *one-time querying*), since users are able to subscribe to information sources and be notified when new events of interest happen. The need for *content-based* push technologies is also stressed by the deployment of new applications (e.g., *Google Alerts*) where a user posts a *subscription* (or *continuous query*) to the system to receive *notifications* whenever matching events of interest take place. Information filtering and information retrieval are often referred as *two sides of the same coin* [BC92]. Although many of the underlying issues and goals are similar in retrieval and filtering, since in both cases a document needs to be matched against an information demand, the design issues, the techniques and algorithms devised to increase filtering efficiency differ significantly. This concept will be one of the main drivers of this thesis.

The main challenge addressed in this thesis is to exploit P2P technology for efficient and approximate information filtering. While there exist several approaches to perform exact information filtering in P2P environments, the work in this thesis emphasizes a novel architecture to support content-based *approximate* information filtering. Most information filtering approaches taken so far have the underlying hypothesis of potentially delivering notifications from *all* information producers.

An approximate approach relaxes this assumption by monitoring only selected sources that are likely to publish documents relevant to the user interests in the future. Since in an IF scenario the data is originally highly distributed residing on millions of sites (e.g., with people contributing to *blogs*), a P2P approach seems an ideal candidate for such a setting. However, exact publish/subscribe functionality has proven expensive for such distributed environments. Contrary, approximate IF offers a natural solution to this problem, by avoiding document granularity dissemination as this presents the main scalability bottleneck for the exact IF approaches.

The targeted P2P system consists of a number n of publisher and/or subscriber *peers* (or *nodes*[1]) d_j with $j = 1, \dots, n$, forming a network. In general, peers are assumed to be autonomous publishing and requesting data. All publisher peers d_j construct and store a local index structure consisting of index lists, $I_j(k)$ over each key k (also referred to as *keyword* or *term*). A continuous query in such a filtering system is a set of multiple distinct keywords $q = k_1, k_2, \dots, t_m$. The subscriber wants to be notified regarding new documents containing these keywords published by other peers in the network. Therefore, all peers have precomputed statistical summaries (metadata) on their local index contents organized on a per-key basis and typically including measures such as the number of documents that the peers's local index contains for a given key, the average term frequency in these documents, and so on. Additionally, these summaries may contain compact synopses representing the documents each peer maintains. These summaries are then posted to a conceptually global, but physically distributed directory conveniently accessible by every peer, with $O(\log n)$ communication costs per key where n is the size of the network. Assuming the existence of a such a directory, the approximate information filtering system has to provide the ability to identify the top-k most appropriate publisher peers $p_1, p_2, \dots, p_k$ for a given continuous query q, i.e., those publishers that are expected to provide the best documents matching q in the future. In this thesis, this task is referred to as *publisher peer selection.* To select this set of peers, this thesis introduces new strategies based on already known *resource selection* techniques in combination with new *behavior prediction* techniques. These new techniques utilize the distributed directory to predict future publishing behavior of peers by applying prediction techniques based on *time series* of IR statistics.

1.2 Main Contributions

This section summarizes the major contributions of this doctoral thesis. Section 1.2.1 discusses the general framework for supporting approximate information filtering whereas Section 1.2.2 extends the framework to consider correlations among keywords. Section 1.2.3 introduces a prototype system, and Section 1.2.4 presents a digital library (DL) use case.

1.2.1 Approximate Information Filtering

This thesis presents a novel architecture, coined MAPS (*Minerva Approximate Publish Subscribe*) [ZTB+08, ZTB+07], to support content-based approximate information filtering in P2P environments. Most IF approaches so far have the underlying hypothesis of potentially delivering notifications from *every* information producer. Contrary, MAPS relaxes this assumption and monitors only selected sources that are likely to publish documents relevant to the user interests in the future. In MAPS, a user subscribes with a continuous query and only published documents from these monitored sources are forwarded to him.

[1]The thesis uses the terminology *peer* rather than *node*.

MAPS provides a network-agnostic P2P architecture with different services and its related protocols (directory, subscription, publication, and notification protocol) for supporting *approximate* IF functionality in a distributed P2P environment. It is the first approach that looks into the problem of approximate IF in such a setting. The architecture exploits metadata stored in a conceptually global, but physically distributed directory. The most critical task in approximate IF is the selection of appropriate publisher peers to meet the information demand in the future. Therefore, MAPS combines existing *resource selection* techniques with new *behavior prediction* strategies. It is shown, that existing resource selection approaches (also referred to as collection or database selection) are not sufficient in a dynamic filtering setting since resource selection can only select appropriate authorities that have already published matching documents in the past.

The novel method of behavior prediction of publisher peers completes the peer selection strategy by applying prediction techniques to *time series* of IR statistics. So, MAPS introduces research of time series analysis to P2P information filtering environments. The experimental evaluation of MAPS approves the effectiveness and efficiency in several settings using real Web data. Various publishing behaviors are investigated to conclude that only the combination of resource selection and behavior prediction allows to improve recall while monitoring only a small number of publishers.

MAPS also conducts an approach to improve the proposed prediction method of *time series analysis* with *double exponential smoothing* (DES). The *MAPS Selective Method* (MSM) does not need any additional communication to adjust prediction parameters. There, the key concept allows to recognize individual publishing behaviors of peers for various continuous queries. In addition, MAPS is compared to an existing exact information filtering approach in the P2P setting with regard to several major characteristics (e.g., load balancing issues, query placement, or routing infrastructure) [TZWK07].

1.2.2 Correlation Awareness

In MAPS, publisher peer selection for a given continuous query with multiple keywords is driven by statistical summaries (metadata) that are stored by the system. These summaries are provided to the directory by the publishers and can be managed in different ways ranging from centralized solutions like servers or server farms, to super-peer or pure peer-to-peer solutions in the form of a distributed P2P directory built on top of a *distributed hash table* (DHT) or other kinds of overlay networks. For scalability of MAPS, the summaries have publisher granularity, not document granularity, thus capturing the best publisher for certain *keywords* or *keys*.

This, together with per-key organization of the directory that disregards keyword correlations[2] or correlated *key sets* are two of the basic reasons that may possibly lead to insufficient recall. Obviously, considering statistics for *all* possible key sets is clearly not possible due to the explosion in the feature space. The baseline approach would decompose a continuous query into the individual keys and use the statistics from the directory to compute a combined score (e.g., intersection or some other kind of aggregation of individual key scores) for each key and publisher. This score would represent the probability of each source to publish documents matching the information demand in the near future. This approach may lead to poor filtering quality as the top-ranked publishers for the *complete* query may not be among the top selected publishers. In the worst case, a selected publisher may deliver many documents for each single keyword, but no single document matching *all keywords*, since this information is not present in the directory.

[2] In general statistical usage, correlation or co-relation refers to the departure of two variables from independence.

Thus, the thesis introduces two approaches as suggested in [ZTW08] that use correlations among keys to improve filtering quality in the scenario described above:

- The *USS* (*Unique Synopses Storage*) algorithm uses existing single-key synopses stored in the directory to estimate the publishing behavior of information sources for key sets.
- The *CSS* (*Combined Synopses Storage*) algorithm enhances the directory to explicitly maintain statistical metadata about selectively chosen key sets.

Contrary to distributed IR settings for one-time searching where sources are ranked according to their document collections (i.e., using resource selection strategies), in approximate IF the publishers are ranked according to their probability to publish relevant documents in the near future, which poses different requirements for maintaining metadata.

This thesis presents the first work to develop algorithms for exploiting keyword correlations in such a dynamic IF setting. Existing and self-limited approaches for two-key queries are extended to the case of multi-key continuous queries for an arbitrary number of keys. Beyond that, new algorithms to approximate multi-key statistics by combining the statistics of arbitrary subsets are provided. Whereas hash sketches, used for compactly representing the documents, yield inaccurate results when considering continuous queries with more than two keys, the usage of very recent state-of-the-art techniques (KMV synopses) for compact representation of multisets is proposed and applied. These new structures allow the system to compute accurate synopses for multi-key queries, and improve the filtering effectiveness.

The experimental evaluation of both algorithms (USS and CSS) illustrates the filtering performance improvements in comparison to the basic MAPS approach. All experimental series use two different real-world collections for Web and blog data, and apply real-world queries from *Google Zeitgeist*. The evaluation also investigates filtering performance gains depending on the introduced correlation measure (*conditional probability*) representing a way to compute the relatedness among keys.

1.2.3 Prototype System

Within this thesis, a prototype implementation [ZHTW08] has been developed. This prototype was initially meant to serve as a testing environment for the conducted experiments, in order to evaluate the novel methods for approximate filtering, but also aimed at demonstrating the feasibility of the combination of retrieval and filtering functionality in a single unifying system.

Thus, the approximate information filtering approach MAPS introduced in this thesis has been integrated into the *Minerva* search prototype [BMT+05b] such that Minerva also provides an approximate publish/subscribe functionality. In this regard, the implementation aspects concerning the extension of Minerva are investigated and some new components (to subscribe with continuous query, and to publish new documents) upgrade the existent prototype. An extensive use case describes the appliance of the extended Minerva prototype by executing sample one-time and continuous queries in detail. There, the various parts of the graphical user interface are illustrated. In this context, a short overview of existing retrieval and filtering prototypes is given.

1.2.4 Digital Library Use Case

The thesis presents a digital library use case (called *MinervaDL*) as an application scenario to support *approximate* information retrieval and filtering functionality in a single unifying framework. MinervaDL, as introduced in [ZTW07], is hierarchical and utilizes a distributed hash table to achieve scalability, fault-tolerance, and robustness in its routing layer. The MinervaDL architecture allows handling huge amounts of data provided by DLs in a distributed and self-organizing way, and provides an infrastructure for creating large networks of digital libraries with minimal administration costs. There are two kinds of basic functionality that are offered in the DL architecture of MinervaDL:

- In an information retrieval scenario (or one-time querying), a user poses an one-time query and the system returns all resources matching the query (e.g., all currently available documents relevant to the requested query).
- In an information filtering scenario (or publish/subscribe or continuous querying), a user submits a continuous query (or subscription) and will later be notified from the DL system about certain events of interest that take place (i.e., about newly published documents relevant to the continuous query).

The proposed DL architecture is built upon a distributed directory storing metadata. The thesis presents routing protocols for information filtering and retrieval that use this directory information to perform the two functionalities. MinervaDL distinguishes three main components:

- *Super-peers* run the DHT protocol and form a distributed directory that maintains metadata about providers' local knowledge in compact form. In MinervaDL, super-peers are responsible for serving information consumers and providers and act as their *access point* to the network. Super-peers can be deployed by large institutions like universities, research centers or content providers to provide access points for their users or digital libraries.
- *Consumers* are utilized by users (e.g., students, faculty or employees) to connect to the MinervaDL network, using a single super-peer as their access point. Utilizing a consumer peer allows users to pose one-time queries, receive relevant resources, subscribe to resource publications with continuous queries and receive notifications about published resources (e.g., documents) that match their interests.
- *Providers* are implemented by information sources that want to expose their content to the MinervaDL network. Typical examples are digital libraries deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer, or Elsevier). Provider peer use a directory peer (super-peer) as their access point and utilize it to distribute statistics about their local resources to the network. Providers answer one-time queries and store continuous queries submitted by consumers to match them against new documents they publish.

This thesis presents different scoring functions to select appropriate provider peers answering a one-time query or storing a continuous query for future matching publications. The experimental evaluation investigates the influence of the individual scoring functions used to perform the two functionalities of MinervaDL. Furthermore, MinervaDL is compared to another DL architecture for P2P networks (*LibraRing*) providing retrieval and filtering at the same time. Unlike MinervaDL, this architecture provides exact retrieval and filtering functionality.

1.3 Thesis Outline

This thesis is comprised of eight chapters, the first being the current introductory chapter. The remainder of this thesis is organized as follows. Chapter 2 gives an background overview about existing work in the areas of P2P, information retrieval, information filtering, time series analysis, and distinct-value estimation. Chapter 3 introduces the main architecture that is used for approximate information retrieval and information filtering in structured P2P networks. This architecture extends the Minerva search architecture with the MAPS approach for approximate information filtering. This chapter also presents the services and protocols of the complete architecture but with focus on approximate filtering. Chapter 4 focuses on publisher peer selection in the approximate information filtering approach MAPS and includes an extensive evaluation. Chapter 5 extends the MAPS approach and considers correlated keys. Two different algorithms that improve filtering quality are presented and compared to the baseline approach. Chapter 6 presents the implemented prototype system for approximate information filtering. This prototype extends the Minerva P2P search prototype allowing one-time searching. In Chapter 7, a DL use case for searching and filtering in a distributed digital library environment is put forward. The proposed DL architecture combines two functionality using the same infrastructure. This thesis is concluded in Chapter 8 by summarizing the main results and pointing out some open questions and directions for future work in this area.

Chapter 2

Background and Technical Basics

This chapter introduces some background and state-of-the-art work used to develop this doctoral thesis. First, a short overview of existing P2P architectures and protocols is described in Section 2.1 including the basic underlying infrastructure. Sections 2.2 and 2.3 present necessary background knowledge about *information retrieval* and *information filtering*.

The main techniques in the area of *time series analysis* are explained in Section 2.4, providing the background for the publication prediction methods. Finally, Section 2.5 presents two data structures (or *synopses*) for distinct-value estimation on large data sets that are used to count the number of distinct data items within a multi-set.

2.1 Peer-to-Peer Systems

In recent years, the *peer-to-peer* approach has been receiving increasing attention and has become a true hype paradigm for communication on the Internet and the World Wide Web. Although becoming popular mainly in the context of file sharing applications (e.g., *Napster*, *Gnutella*, or *BitTorrent*), the P2P paradigm can be applied to access any kind of distributed data. Currently, P2P is making its way into distributed data management systems and is offering possibilities for new Web applications.

In contrast, the traditional client-server approach (shown in Figure 2.1) requires a huge amount of effort and resources to meet the increasing challenges of the continuously growing resources shared over the Internet. This centralized client-server network model results in increased load for the server component, and creates scalability bottlenecks and *single-points-of-failure*, where a failure of one entity results in terminating the functionality of the whole system. As a consequence, such systems can easily be attacked, e.g., by *denial-of-service* attacks. In addition, dedicated servers are often difficult and expensive to administrate and to relocate due to their strategic placement within the Internet infrastructure.

Contrary to the client-server model, P2P computing promises to offer enormous potential benefits to important issues including scalability, security, reliability, efficiency, flexibility, and resilience to failures and dynamics. [SW05] discusses in detail the issues addressed by this fundamental paradigm shift.

An exact answer to the question *What exactly is P2P?* is not obvious. Especially the early P2P applications that have not even been true P2P in a strict sense, make it difficult to give a precise definition. The Internet encyclopedia *Wikipedia* currently defines the following aspects of a P2P network:

- *A peer-to-peer (or "P2P", or, rarely, "PtP") computer network exploits diverse connectivity between participants in a network and the cumulative bandwidth of network*

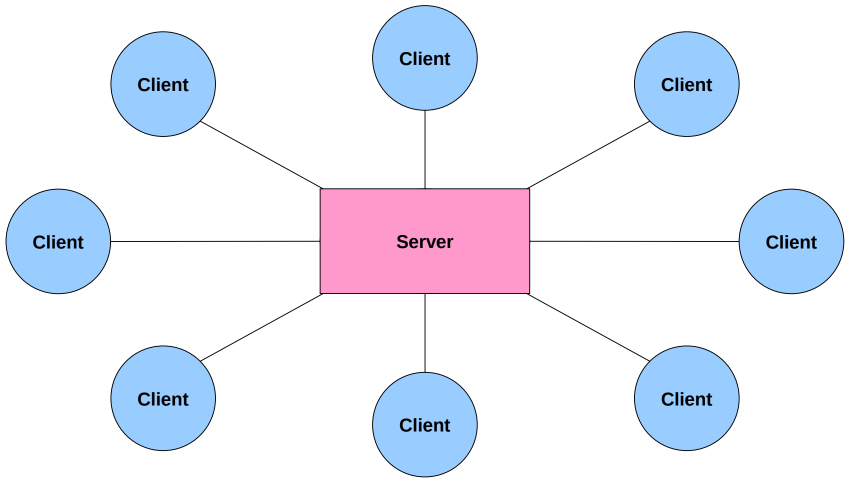


Figure 2.1: Client-Server Network Architecture.

participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application.

- *Peer-to-peer networks are typically used for connecting nodes via largely ad hoc connections.*
- *A pure peer-to-peer network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server.*

A more technical definition of P2P systems (shown in Figure 2.2) can be found in [Ora01, SW05]: A peer-to-peer system is *a self-organizing system of equal, autonomous entities (peers)* which *aims for the shared usage of distributed resources in a networked environment avoiding central services.* Both definitions share the idea of decentralization and point at decentralized resource usage and decentralized self-organization as potential benefits.

2.1.1 The Lookup Problem

How to find a certain data item in a large P2P system in a scalable manner without any centralized service? This problem is at the heart of any P2P system [BKK+03] and is often referred to as the *lookup problem*. The main reason of this challenge is caused by decentralization such that the main system strength also yields to the main system challenge.

In contrast to centralized client-server systems, where data is provided by dedicated physical entities that are explicitly referenced, P2P systems store data in multiple, distant, transient, and unreliable locations within the network. Thus, the efficient location of data stored in the network is one of the predominant challenges of a P2P system.

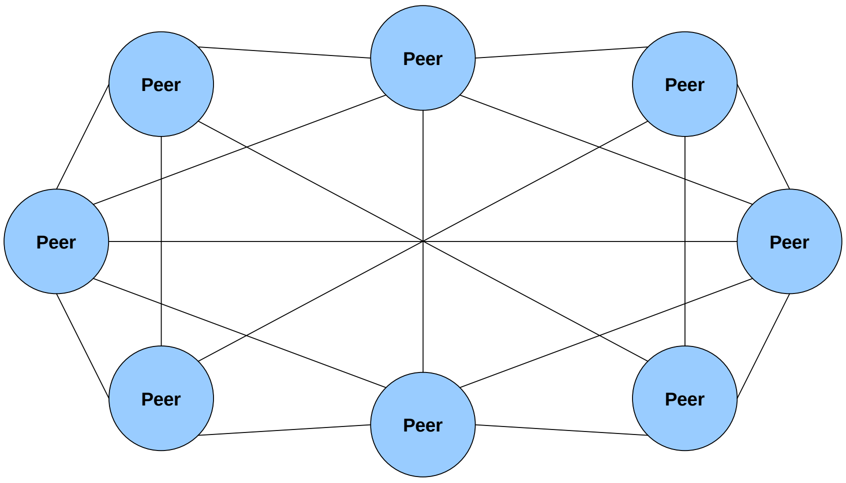


Figure 2.2: Peer-to-Peer Network Architecture.

2.1.2 P2P Architectures

There are several ways to classify P2P networks. One approach considers the application a P2P network is used for (e.g., file sharing, telephony, media streaming etc.). Another approach includes the degree of centralization and distinguishes between *pure* P2P without central server (peers act as equals) and networks with central server keeping information on peers. In this context, the following terminology can be found: *centralized*, or *decentralized* P2P networks, *structured*, *unstructured*, or *hybrid* (so-called *super-peer* architectures) P2P networks.

The research literature commonly tries to classify the existing approaches to deal with the *lookup problem*. The following sections present the respective approaches in more detail. Subsequent, two example protocols of P2P architectures are presented: *Chord* (2.1.3) and *Pastry* (2.1.4).

2.1.2.1 Centralized P2P Architectures

The first occurrence of the P2P paradigm in a broader public cognition was probably *Napster*[1]. This *famous* file-sharing system elegantly solved the lookup problem by utilizing an architecture in which a centralized entity provides a directory service to all participating peers (or users), effectively forming a *star network*. All peers joining the system have to register their data (mostly music files in the early days) with this centralized server thus allowing a comfortable way for other peers in the system to locate any data in the network by presence of a physically centralized directory. The fact that only pointers to decentralized available peers are stored at the centralized server (instead of the actual data) conceptually decreases the load at the central entity; the fact that (after relevant data has been located by means of the directory) each peer could directly communicate with other peers that store the data in a decentralized manner, completely bypassing the centralized directory entity, drives the perception of Napster as a P2P system.

[1]Napster's brand and logo continue to be used by a pay service, having been acquired by *Roxio*.

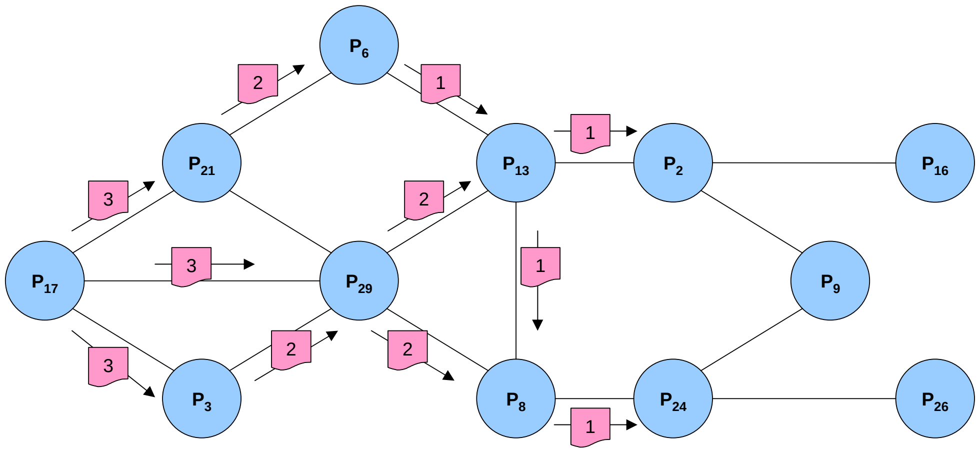


Figure 2.3: Unstructured P2P Network with Message Flooding.

However, in a *pure* P2P system, it should be possible to remove any entity from the network without loss of functionality. Instead, a peer should conceptually fulfill both roles as server and client, such that the functionality of the system is spread equally over all the participating peers in the network. Thus, by this definition, Napster can not be denoted as a P2P system. The shutdown of the the centralized server of Napster by legal authorities allowed the easy shutdown of the complete system.

2.1.2.2 Unstructured P2P Architectures

In contrast to centralized approaches, *non-centralized* or *decentralized* P2P architectures do not rely on any centralized or coordinating entity to locate data items within the network. More specifically, in *unstructured* P2P approaches which are a specialization of the decentralized architectures, peers recursively forward received requests to neighboring peers (also referred to as neighbors), in an attempt to find all relevant items in the network. In order to avoid missing peers in the network, each peer broadcasts messages to all known other peers, regardless of whether these neighbors store relevant data or not. This message forwarding approach leads to a *breadth-first search* strategy and is also known as *message flooding*. To prevent infinite loops and to control the number of messages generated by one single request, each message gets assigned a *time-to-live* (TTL) value. Each peer forwarding such a message decreases this value by one, and only messages with positive TTL values get forwarded.

Figure 2.3 illustrates message flooding in an unstructured P2P network architecture. The peer on the left issues a query and forwards this request message to its three known neighbors, as indicated by the arrows. As shown, the initial TTL value of the query message is three. After decreasing the TTL value, these peers forward the message to their own neighbors. If the TTL value is not positive after decreasing it, the peer will not forward the message further, thus terminating the forwarding procedure. Note that some messages unnecessarily address peers that have already received the query before (e.g., from different peers), but peers do not forward the same message to its neighbors more than once.

There are several studies of real-world networks showing that the peers of such a network form graphs with small diameters, typically in a range from five to seven, supporting the so-called *small-world-phenomenon* [Kle00], a property that has also been observed in *social networks*[2]. One important property of such networks is that messages with a relatively small TTL value can locate any requested data with high probability. But, this technique still represents a lossy protocol as it has no guarantees to successfully locate the data of interest, e.g., due to higher graph diameters or disconnected graph partitions. Additionally, the number of messages created by a single request is significant; it depends on the degree of the peers in the network (i.e., the number of neighbors of a peer) and the chosen TTL value.

The main advantage of unstructured P2P networks is the fact that there is no need to proactively maintain a network structure. Peers maintain only pointers to an upper-bounded number of direct neighbors. Also note that there is no enforcement of a specific storage location for data items, as they can be located anywhere in the network. In other words, the data stored at a peer is unrelated to the peer's position in the network. *Freenet* [CMH+02] and early versions of the *Gnutella* protocol[3] are popular representatives of this paradigm. In unreliable networks (e.g., mobile ad-hoc networks), flooding is also a fundamental message dissemination strategy. While *brute-force flooding* algorithms cause a high number of unnecessary messages, network contention, packet collisions, and wasting energy, several probabilistic and epidemic approaches have been studied to optimize plain flooding.

2.1.2.3 Structured P2P Architectures

Centralized P2P architectures suffer from scalability bottlenecks which prevent an a-priori unlimited number of participating peers. In a centralized architecture, the linear storage complexity of the central directory entity is prohibitive. In an unstructured architecture, the communication overhead caused by message flooding is a significant deficit. Thus, an efficient and scalable approach requires a sub-linear increase in the storage and search complexity as more and more peers join the network.

Structured P2P architectures utilize particular overlay structures to map peers and data items into a common address space, enabling a unique mapping from data items to peers given the current state of the network. Therefore, each peer manages a small number (typically $O(\log N)$, where N is the number of peers in the whole network) of pointers to carefully selected other peers. If these structured overlays are used for routing, then reaching the peer currently responsible for a given data item requires $O(\log N)$ messages. To guarantee balanced storage and retrieval loads among the peers, the responsibilities for data items have to be distributed as uniformly as possible.

To realize this routing functionality, a distributed data structure based on a hash table is used to allow the insertion and retrieval of *key/value* pairs. Thus, to insert or retrieve a *key/value* pair, the peer responsible for a *key* in the network as defined by the structured P2P network is utilized. This peer stores and maintains all appropriate *key/value* pairs for a *key* from across the directory. Note that, in contrast to unstructured P2P architectures, the data placement is no longer arbitrary; the placement is accurately determined by the underlying overlay architecture, and this provides a guarantee to find data items indexed by the network.

[2] A social network is a social structure made of nodes (which are generally individuals or organizations) that are tied by one or more specific types of interdependency.

[3] Gnutella is a file sharing network. In December 2005, it was the third-most-popular file sharing network on the Internet.

The expression *distributed hash table* stands for such a functionality in a P2P network and *DHT* is commonly used as a synonym for structured P2P architectures in general. But, there is also the strict distinction between structured P2P routing primitives on one side and the DHT interfaces of inserting and retrieving data as upper layer of functionality on the other side. Throughout this thesis, DHT is used to refer to the class of structured P2P architectures.

There are several proposals for structured overlay topologies including geometries as hypercubes (e.g., *CAN* [RFH+01]), rings (*Chord* [SMK+01], or *Pastry* [RD01a]), tree-like structures (*P-Grid* [Abe01], or *PRR* [PRR97]), and butterfly networks (*Viceroy* [MNR02]). Special cases are random graphs [MS05]. [GGG+03] presents a study concerning the general resilience and proximity properties of these different geometries.

2.1.2.4 Super-Peer Architectures

Super-peer architectures exploit the fact that the performance characteristics of the peers (processing power, bandwidth, availability, etc.) is not evenly distributed over all peers in the network. Thus, the benefits of a perfect decentralization are decreasing.

In super-peer architecture, a small subset of peers takes over specific responsibilities in the network, e.g., aggregation or routing tasks. Thus, the *super-peers* can be viewed as the distributed extensions of the centralized entity in the Napster architecture. Conceptually, only the super-peers build-up the P2P network; all other peers connect to this backbone by communicating to one super-peer, which acts in the spirit of database mediators aggregating the content of downstream peers.

Routing in super-peer architectures is conducted in a two-phase mode. A request is routed within the super-peer backbone at first, and is then distributed to the peers connected via the super-peers. While dedicating specific peers potentially limits the self-organizing capabilities of a P2P network, super-peer architectures have been proven a way to alleviate the performance issues of pure unstructured topologies. [YGM03] examines super-peer networks in detail to gain an understanding of their fundamental characteristics and performance tradeoffs. In Chapter 7, this thesis uses a super-peer architecture to build-up a distributed *digital library* environment.

2.1.3 Example: The Chord Protocol

The *Chord* protocol [SMK+01] is definitively one of the most prominent DHT implementations and one of the most important representatives of structured P2P architectures. The main advantage of Chord when compared to other DHT approaches is its elegance and simplicity of concepts and implementation, and its still growing popularity.

In Chord, all data items and peers are mapped to a unique one-dimensional identifier space such that identifiers are l-bit numbers, i.e., integer values in the range $[0, 2^l - 1]$, forming a cyclic identifier space modulo 2^l. In the following, the identifier of a data item is referred to as a *key*, while the identifier of a peer as an *id*. The responsibility of maintaining the data items $(key, value)$ associated with *key* lies at the nearest peer on the *identifier circle* (*ID circle* or *Chord ring*) whose *id* is greater or equal to *key*. This peer p is called the *successor* of *key*. Thus, a peer p is responsible for all *key* identifiers between its *predecessor*'s identifier (exclusive) and its own identifier (inclusive), and each *key/value* pair is located and managed on a single, well-defined peer.

Figure 2.4 illustrates an identifier circle with $l = 6$, i.e., identifiers in the range $[0, 63]$. For example, key K_{54} is maintained by peer P_{56} as its next successor on the ID circle, whereas both keys K_{24} and K_{30} are maintained by peer P_{32}.

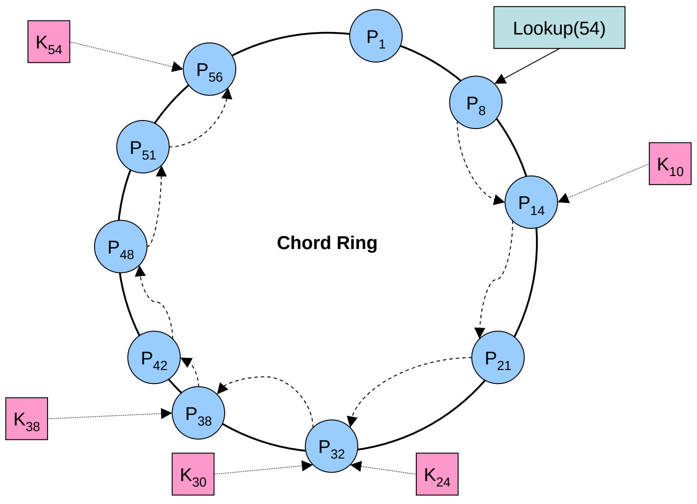
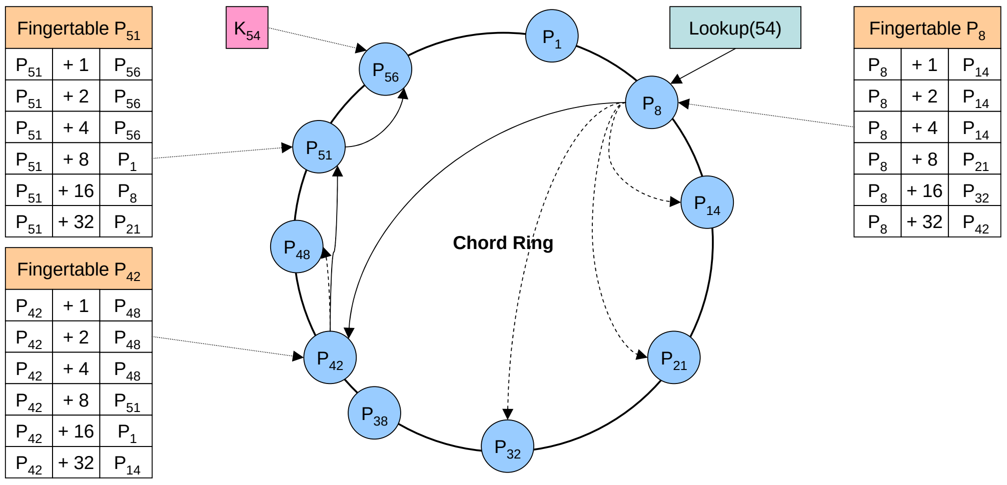


Figure 2.4: Example Chord Ring.

The solution to efficient lookup and modification operations on the stored data is to efficiently solve the *lookup problem* introduced in 2.1.1, i.e., to quickly locate the peer responsible for a particular *key*. As a naive approach, every peer could store a direct pointer to its successor peer on the identifier circle. When a key is being requested, each peer forwards the request to its successor, until one peer determines that the key lies between its own *id* and the *id* of its successor. Thus, the key must be hosted by its successor. As a consequence, contact information about the successor is communicated as the result of the query back to the originator. When maintaining only a minimum amount of state at each peer ($O(1)$), this form of key location leads to an expected number of messages linear in the number of peers in the network, which is not considered scalable and acceptable. Figure 2.4 also shows this naive approach where peer P_8 issues a lookup request for key K_{54}. The request is forwarded along the ID circle linearly until the responsible peer P_{56} can be identified.

To improve scalability and efficient lookups, Chord keeps additional state at each peer. A peer maintains a routing table (also referred to as *finger table*), pointing to other peers on the identifier circle. The m-th entry in the finger table of peer P_i contains a pointer to the first peer P_j that succeeds P_i by at least 2^{m-1} on the identifier circle, leading to a finger table with at most l distinct entries (independent of the actual number of keys or peers).

There are two important characteristics of this schema. First, each peer only maintains state about a logarithmic number of other peers, and knows more about peers closely following it on the identifier circle than about peers farther away. Second, a peer's finger table does not necessarily contain enough information to directly determine the peer responsible for an arbitrary *key*. However, since each peer has finger entries at power of two intervals around the identifier circle, each peer can forward a query at least halfway along the remaining distance between itself and the requested peer. This property is illustrated in Figure 2.5 for peers P_8, P_{42}, and P_{51}. It follows that the number of peers to be contacted (and, thus, the number of messages to be sent) to find a target peer in an N-peer system is $O(\log N)$.



Fingertable P$_{51}$		
P$_{51}$	+ 1	P$_{56}$
P$_{51}$	+ 2	P$_{56}$
P$_{51}$	+ 4	P$_{56}$
P$_{51}$	+ 8	P$_1$
P$_{51}$	+ 16	P$_8$
P$_{51}$	+ 32	P$_{21}$

Fingertable P$_{42}$		
P$_{42}$	+ 1	P$_{48}$
P$_{42}$	+ 2	P$_{48}$
P$_{42}$	+ 4	P$_{48}$
P$_{42}$	+ 8	P$_{51}$
P$_{42}$	+ 16	P$_1$
P$_{42}$	+ 32	P$_{14}$

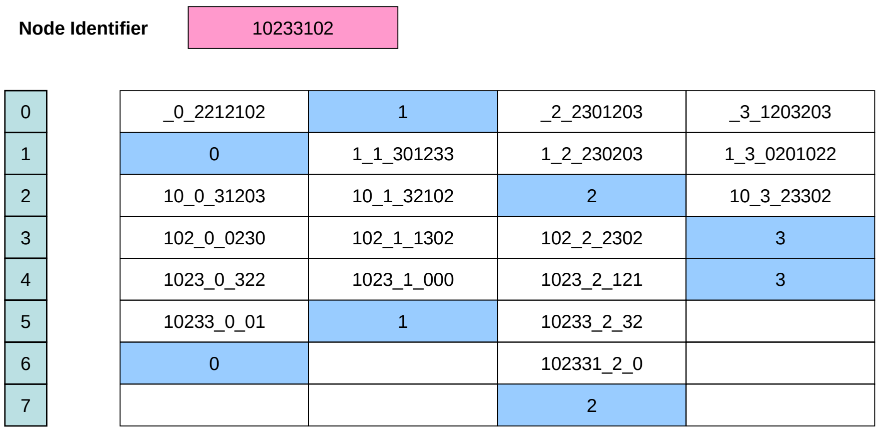
Fingertable P$_8$		
P$_8$	+ 1	P$_{14}$
P$_8$	+ 2	P$_{14}$
P$_8$	+ 4	P$_{14}$
P$_8$	+ 8	P$_{21}$
P$_8$	+ 16	P$_{32}$
P$_8$	+ 32	P$_{42}$

Figure 2.5: Chord Lookup using Finger Table Entries.

To handle *churn*[4] in a network, Chord implements a *stabilization protocol* that each peer runs periodically in the background and which updates the finger tables and successor pointers in order to ensure that lookups execute correctly. The stabilization requires an additional predecessor pointer, as each peer requests its successor, $Succ(P_i)$, to return its predecessor $Pred(Succ(P_i))$. If P_i equals $Pred(Succ(P_i))$, P_i and $Succ(P_i)$ agree on being each other's respective predecessor and successor. In contrast, the fact that $Pred(Succ(P_i))$ lies between P_i and $Succ(P_i)$ indicates that $Pred(Succ(P_i))$ recently joined the identifier circle as P_i's successor. Thus, P_i updates its successor pointer to $Pred(Succ(P_i))$ and notifies $Pred(Succ(P_i))$ of being its predecessor. At this stage, all successor pointers are up to date and requests can be routed correctly. As the impact of outdated finger table entries on lookup performance is small, Chord updates finger tables only lazily by periodically picking a finger table entry j randomly and performing a lookup to find the true peer that currently succeeds peer P_i by 2^{j-1}.

Chord addresses peer failures by checking all communication with remote peers for time-outs. To further ensure routing stability in the presence of multiple simultaneous peer failures, each peer maintains not only a pointer to its immediate successor, but a list of the first r successors. When a peer detects a failure of its successor, it reverts to the next peer in its *successor list*. The successor list is also maintained by the stabilization protocol. [LNBK02] gives a theoretical analysis of Chord's stability in the face of concurrent joins and multiple simultaneous peer failures. The failure of a peer not only puts the routing stability at risk, but also makes the data managed by this peer unavailable. Such data loss can be prevented by replicating the data items to other peers. In Chord, the successor of a failed peer becomes responsible for the keys and data of the failed peer. Thus, an obvious replication strategy to prevent data loss is to replicate data to immediate successors, using the successor list.

[4]The churn rate refers to the number of peers joining and leaving the system during a given period as a significant problem for large-scale systems.



Node Identifier: 10233102

0	_0_2212102	1	_2_2301203	_3_1203203
1	0	1_1_301233	1_2_230203	1_3_0201022
2	10_0_31203	10_1_32102	2	10_3_23302
3	102_0_0230	102_1_1302	102_2_2302	3
4	1023_0_322	1023_1_000	1023_2_121	3
5	10233_0_01	1	10233_2_32	
6	0		102331_2_0	
7			2	

Figure 2.6: Pastry Routing Table.

2.1.4 Example: The Pastry Protocol

Pastry[5] [RD01a] is another self-organizing structured overlay network. The protocol uses a routing schema based on *prefix matching*. Each Pastry node is assigned a globally unique 128-bit identifier from the domain $[0, 2^{128} - 1]$, in form of sequences of digits with base 2^b where b is a configuration parameter. A typical value for b is 4. Similar to Chord, Pastry offers a simple routing method that efficiently determines the node that is numerically closest to a given key, i.e., which is currently responsible for maintaining this key. To enable efficient routing within a network of N nodes, each peer maintains a routing table that consists of $\lceil log_{2^b} N \rceil$ rows with $2^b - 1$ entries each. Each entry consists of a Pastry identifier and the corresponding contact information (i.e., IP address, port) of the numerically closest node currently responsible for that key. All $2^b - 1$ entries in row n represent nodes with a Pastry identifier that shares the first n digits with the current node, but each with a different $n + 1$-st digit ($2^b - 1$ possible values). Figure 2.6 shows an example routing table with 8 rows for a Pastry node with identifier 10233102 and $b = 2$.

The prefix routing now works as follows: For a given key, the current node forwards the request to that node from its routing table that has the longest common prefix with the key. Intuitively, each routing hop can fix one additional digit toward the desired key. Thus, in a network of N nodes, Pastry can route a message to a currently responsible node with less than $\lceil log_{2^b} N \rceil$ message hops.

Pastry is intended as general substrate for the construction of a variety of P2P applications like global file sharing, file storage, group communication and naming systems. Several application have been built on top of Pastry to date, including a global, persistent storage utility called *PAST* [RD01b] and a scalable publish/subscribe system called *Scribe* [RKCD01].

[5]http://freepastry.org/

2.2 Information Retrieval

Information retrieval is the science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases. IR systems keep collections of unstructured or semi-structured data (e.g., text documents or HTML pages) and offer search functionalities for delivering documents relevant to a query. Typical examples of IR systems include Web search engines such as *Google*, *Yahoo* or *Live Search* (formerly *MSN Search*), or digital libraries. Lately, relational database systems are integrating IR functionality as well (see [MYL02, Cha02] for more details about database systems that support IR functionalities).

2.2.1 Effectiveness Measures

Precision and *recall* are the two most widely used measures for evaluating the quality of results in the area of IR. Precision can be seen as a measure of *exactness* or *fidelity*, whereas recall is a measure of *completeness*. Both measures are defined as follows:

$$precision = \frac{|R \cap S|}{|R|} \tag{2.1}$$

$$recall = \frac{|R \cap S|}{|S|} \tag{2.2}$$

In Equations 2.1 and 2.2, S is the set of *retrieved documents* (e.g., the list of documents produced by a Web search engine), and R is the set of all *relevant documents* (e.g., all Web documents relevant for a certain topic).

In an IR scenario, precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by this action, and recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents (which should have been retrieved). A perfect precision score of 1.0 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved) whereas a perfect recall score of 1.0 means that all relevant documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved).

Depending on the selected application, precision or recall have a dominating importance. Applications not accepting irrelevant documents will tend to reduce the number of retrieved documents at the cost of sacrificing recall. Other applications focus on getting all relevant documents, and will tend to increase the number of retrieved documents at the expense of lower precision.

A popular measure that combines precision and recall is the weighted harmonic mean of both, the traditional *F-measure* or balanced *F-score*:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.3}$$

This measure is also called F_1*-measure* because it equally weights both precision and recall in one single parameter. The F_α*-measure* generalizes the *F-measure* and gives one of the two measure a higher weight:

$$F_\alpha = (1 + \alpha) \cdot \frac{precision \cdot recall}{\alpha \cdot precision + recall} \tag{2.4}$$

The F_1-*measure* is a good candidate for optimization. Given the fact that one can get a perfect precision score by not returning any documents, or a perfect recall score by returning all documents. A powerful document retrieval system will yield all truly relevant documents and only truly relevant documents, maximizing precision and recall at the same time, and therefore maximizing the F_1-*score* which falls in the range from 0 to 1 (with 1 being the best possible score). Two other commonly used F-measures are the F_2-*measure*, which weights recall twice as much as precision, and the $F_{0.5}$-*measure*, which weights precision twice as much as recall. More about IR measures can be found in [BY99].

2.2.2 Document and Query Representation

Usually, documents are represented by the *words* (or *terms*, *keywords*, or *keys*) that are contained in them. Typically, a small set of so-called *stop words* are not considered and are ignored because these words do not contain any semantic information (e.g., *a*, *the*, *of*, etc.). *Stemming* describes the reduction of words to their common prefix *stem* thus combining combining words with a common *root* (e.g., *walk*, *walking*, *walker*). A very widely used stemmer that became the de-facto standard algorithm used for English words is the *Porter Stemmer* [RvRP80].

After the removal of all stop words and after stemming, a document in a collection can be represented by a vector of n key occurrences, where n is the total number of distinct keys in the set of all documents in a collection. In this *vector space model*, a document is represented by a vector $(w_1, w_2, \ldots, w_n)$, where each w_i is the weight (or score) indicating the importance of key i in a document. Due to the high number of distinct keys in the whole collection and due to the absent of most keys in a typical document, most of the vector entries will be zero. A weight assigned to a key is usually based on the following two factors:

- The number of occurrences of key t in document d is called *term frequency* and typically denoted as $tf_{t,d}$. Intuitively, the weight of a key within a document increases with the number of occurrences.

- The number of documents in a collection that contain a key t is called *document frequency* and denoted as df_t; the *inverse document frequency* idf_t is defined as the inverse of df_t. Intuitively, the relative importance of a key decreases as the number of documents that contain this key increases, i.e., the key offers less differentiation between the documents.

In practice, these two measures may be normalized (e.g., to values between 0 and 1) and dampened using logarithms. Eventually, multiplying these two values yields the weight of a key in a document. This whole family of approaches is often referred to as $tf - idf$; a typical representative calculates the weight $w_{i,j}$ of key t_i in document d_j of a collection with a total number of N documents as follows:

$$w_{i,j} = \frac{tf_{i,j}}{max_t\{tf_{t,j}\}} \cdot \log \frac{N}{df_i} \tag{2.5}$$

Recently, other relevance measures based on statistical *language models* [PC98, SJCO02] and probabilistic IR have been investigated [Fuh99]. [Cha02, MS99] give a good overview over various relevance measures.

A query consists of a set of (possibly weighted) keys that can again be represented by the appropriate n-vector. The *scalar product* can be applied to compare the similarity between a query q and a document d as follows:

$$scalar(q, d) = \sum_{k=1}^{n} q_i \cdot d_i \tag{2.6}$$

In Equation 2.6, q and d are the query and document vectors, respectively. For a particular query, the scalar product intuitively sums up the importance scores for those keys of the document that are contained in the query vector, weighted by their respective weights within the query. The *cosine measure* can be used to overcome the tendency of the scalar product function to favor longer documents having many keys, capturing the angle between the two vectors by using the vector lengths:

$$cosine(q, d) = \frac{scalar(q, d)}{|q| \cdot |d|} \tag{2.7}$$

An interesting side effect is that, for non-negative weights, the cosine function returns values between 0 and 1, making an additional score normalization unnecessary. *Okapi BM25* is another ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework. BM25 is a *bag-of-words* retrieval function that ranks a set of documents based on the query keys appearing in each document, regardless of the inter-relationship between the query keys within a document. It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. [RWHB+92] describes the most prominent instantiations.

2.2.3 Top-k Query Processing

Computing the scalar product, the cosine measure, or the BM25 score for all documents concerning a query is a computationally expensive task. Thus, there are several strategies to efficiently identify only the top-k documents (i.e., the k documents with the highest score for the given query) without evaluating the full scalar product for all possible documents. In order to efficiently support this process, the concept of *inverted index lists* has been developed [ZM06]. All keys that appear in the whole collection form a tree-based index structure (often a B^+-tree or a trie) where the leaves contain a list of unique document identifiers for exactly those documents that contain this key (see Figure 2.7). Depending on the exact query execution strategy, the lists of document identifiers may be ordered according to the document identifiers or according to their key scores to allow for efficient pruning.

An efficient algorithm should avoid reading inverted index lists completely, but would ideally limit the effort to $O(k)$ steps where k is the number of desired results. In the IR and multimedia-search literature, there are various algorithms to accomplish this task. The *threshold algorithm* (TA) by Fagin [FLN01] (independently proposed also by Nepal et al. [NR99] and Güntzer et al. [GBK00]) is the best known method for top-k queries. It uses index lists that are sorted in descending order of key weights under the additional assumption that the final score for a document is calculated using a monotone aggregation function (such as a simple sum function). TA traverses all inverted index lists in a *round-robin* manner, i.e., lists are mainly traversed using sorted accesses. For every new document d encountered, TA uses random accesses in the remaining index lists to calculate the final score for d and keeps this information in a document candidate set of size k. Since TA additionally keeps track of an upper score bound for documents not yet encountered, the algorithm terminates as soon as this bound assures that no previously unseen document can enter the candidate set.

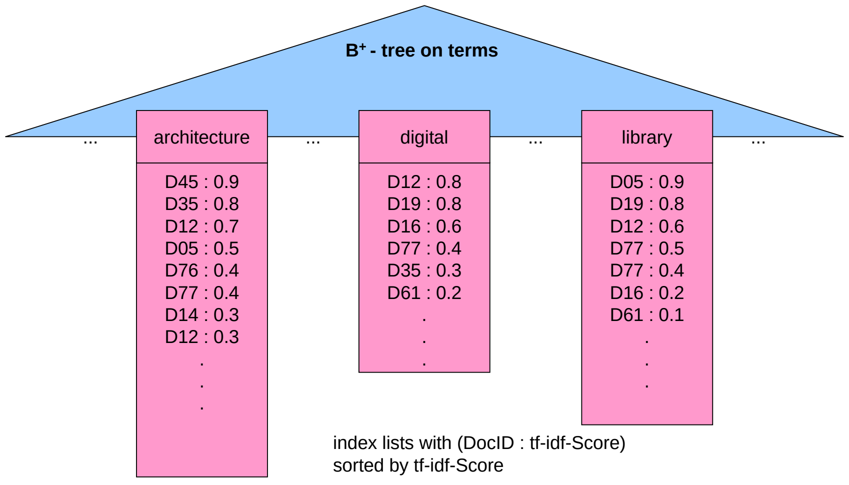


Figure 2.7: B^+-Tree with Inverted Index Lists.

There are extensions of this algorithm that further reduce the number of index list accesses (especially expensive random accesses) by a more sophisticated scheduling. Also, probabilistic methods have been studied that can further improve the efficiency of index processing [TSW05, BMS$^+$06].

2.3 Information Filtering

Information retrieval and *information filtering* (or *selective dissemination* of information or *publish/subscribe*) are often referred as *two sides of the same coin* [BC92]. Although many of the underlying issues and goals are similar in retrieval and filtering, since in both cases a document needs to be matched against an information demand, the design issues, the techniques and algorithms devised to increase filtering efficiency differ significantly. Information filtering can also be seen as an general application of information retrieval because most of the issues which appear at first to be unique to IF, are really specializations of IR problems.

2.3.1 IF in Databases and Distributed Systems

Selective dissemination of information has its origin in a 1958 article about *Business Intelligence System* [Luh58] where a selection module (*selective dissemination of new information*) is used to produce lists of new documents matching interests of users described by profiles. The concept of *information filtering* was introduced in [Den82], where the need to filter incoming mail messages to sort them in order of urgency is described[6]. In the following, only work most relevant to this thesis and mainly referred as *content-based filtering* will be discussed. Early approaches to information filtering by IR researchers focused mainly on appropriate representations of user interests [MS94] and on improving filtering effectiveness [HPS96].

[6]At that time, spam filters were not needed since spam was not such a big issue than today.

[BM96] is one of the first approaches to address performance, where an information filtering system capable of scaling up to large tasks is described. Here, a server receives documents at a high rate, and proposed algorithms support *vector space queries* by improving the algorithm *SQI* of [YGM94a]. *InRoute* [Cal96] creates documents and query networks and uses belief propagation techniques to filter incoming documents based on inference networks. [ZC01, Cal98] mainly focus on *adaptive filtering* and how vector space queries and their dissemination thresholds are adapted based on documents processed in the past. Filtering news articles [CGR05, GDH04] is also an important research area related to IF but stresses the focus on personalized *duplicate elimination* (or information novelty) and freshness of the results shown to the user.

In the database literature, the term *selective dissemination of information* is used [FZ98]. The term *publish/subscribe system* (or *pub/sub system*) comes from distributed systems, and has also been used in this context by database researchers. *SIFT* [YGM99, YGM94b] constitutes another influential system where publications are documents in free text form and queries are conjunctions of keywords. This system was the first approach to emphasize query indexing as a means to achieve scalability in pub/sub systems [YGM94b]. Later, other work focused on pub/sub systems with data models based on attribute-value pairs and query languages based on attributes with arithmetic and string comparison operators [FJL+01, NACP01]. [CCC+01] considers a data model based on attribute-value pairs but goes beyond conjunctive queries (the standard class of queries). Subscription summarization to support pub/sub functionality was also a prominent example [TE04].

Recent work on XML data management [KP05] has focused on publications that are XML documents and queries that are subsets of XPath or XQuery (e.g., *XFilter* [AF00], *YFilter* [DAF+03], or *Xtrie* [CFGR02]). All these papers discuss sophisticated filtering algorithms that base on indexing queries. There are several publish/subscribe systems that have been developed over the years in the area of distributed systems and networks. Data models are based on channels, topics, and attribute-value pairs [CRW01]. The latter systems are called content-based as in the IR literature, because attribute-value data models are flexible enough to express the content of messages in various applications. The query language of content-based systems is based on Boolean expressions of arithmetic and string operations. The *SIENA* system [CRW01] uses a data model and language based on attribute-value pairs and demonstrates how to express notifications, subscriptions and advertisements in this language.

2.3.2 IF in Peer-to-Peer Systems

The two pub/sub systems *DIAS* [KKTR02] and *P2P-DIET* [KTID03] use some prominent ideas from the database and distributed systems in a single unifying framework. The P2P-DIET [IKT04b, IKT04a] approach demonstrates how to provide the traditional one-time query scenario of typical super-peer systems and the pub/sub features of the SIENA system [CRW01]. An extension of P2P-DIET considers a similar problem for distributing RDF meta-data in an *Edutella* [NWQ+02] fashion. The development of structured P2P systems mainly based on *distributed hash tables* such as *CAN* [RFH+01], *Chord* [SMK+01], *Pastry* [RD01a]), *P-Grid* [Abe01], or *Tapestry* [ZZJ+01] introduced a wide-area of new pub/sub systems. *Scribe* [RKCD01] is a topic-based publish/subscribe system based on Pastry. *Hermes* [PB02] is similar to Scribe because it uses with Pastry the same underlying DHT but it allows more expressive subscriptions by supporting the notion of an event type with attributes. Every event type in Hermes is managed by an event broker which is a rendezvous node for subscriptions and publications related to this event. Related ideas and approaches appear in [TAJ04, TBF+03].

The *PeerCQ* system [GL03] is another important pub/sub approach using a DHT infrastructure. PeerCQ takes into account peer heterogeneity and extends consistent hashing [KLL+97] with simple load balancing techniques based on appropriate assignment of peer identifiers to network peers. The *Meghdoot* system [GSAA04] is a content-based pub/sub approach implemented on top of a CAN-like DHT, and supports an attribute-value data model and new ideas for the processing of subscriptions with range predicates (e.g., the price is between 50 and 100 Euros) and load balancing. *Mercury* [BAS04] also uses an attribute-value data model similar to Meghdoot and is utilized in the implementation of a pub/sub system for network games. *PastryStrings* [AT06] also supports a rich set of queries in the context of a pub/sub system. The system utilizes prefix-based routing to facilitate both numerical and string queries.

The *pFilter* system [TX03] uses a hierarchical extension of CAN DHT to filter unstructured documents. This approach relies on multi-cast trees to notify subscribers. VSM and LSI can be used to match documents to user queries. The multi-cast trees of pFilter take into account network distance. Supporting prefix and suffix queries in string attributes is the focus of the *DHTStrings* system [AT05]. This system utilizes a DHT-agnostic architecture to develop algorithms for efficient multi-dimensional event processing. *LibraRing* [TIK05a] was the first approach to provide protocols for the support of both IR and IF functionality in digital libraries (DLs) using DHTs. In LibraRing, super-peers are organized in a Chord DHT and both (continuous) queries and documents are indexed by hashing words contained in them. This hashing scheme depends heavily on the data model and query language adopted, and the protocols have to be modified when the data model changes. The DHT is used to make sure that queries meet the matching documents (in the IR scenario) or that published documents meet the indexed continuous queries (in the IF scenario). In this way the retrieval effectiveness of a centralized system is achieved (in contrary to the work in this thesis), while a number of routing optimizations (such as value proxying, content based-multicasting, etc.) are used to enhance scalability. The filtering approach of LibraRing is presented in detail in [TIK05b, TKD04, TKD08].

2.4 Time Series Analysis

In the literature (e.g., [Cha04],or [Ham94]), a *time series* is declined as a collection of observations made sequentially through time. Time series play an important role in a huge variety of fields ranging from economics to engineering. The different methods to analyze and explain time series constitute an notable area of statistics and this area is called *time series analysis*.

Many time series arise in practice, for instance by recording values for a period of time. Prominent examples of time series in economics and finance are export totals in successive years, or average incomes in successive months. [Cha04] mentions the classic Beveridge wheat price index series consisting of the average wheat price in nearly 50 places in many countries in successive years from 1500 to 1869. Besides economic and financial time series, there are a lot of other important areas: *physical time series*[7], *marketing time series*, and *demographic time series*[8]. Time series analysis also occurs in *process control* to ensure a certain quality level. *Binary processes* are a special type of time series where observations can take only one of two values (usually denoted by 0 and 1). Communication theory represents a situation of this type of time series.

[7]Including meteorology, marine science and geophysics (e.g., rainfall on successive days).

[8]Time series study the population development of countries considering among other things birth and death rates.

The literature also considers events occurring *randomly* through time[9]. A series of events of this type is called *point process*. There is a big difference between analyzing point processes and analyzing standard time series data as the interested quantities differ (e.g., the distribution of the number of events occurring in a given time period).

2.4.1 Terminology

A time series is called *continuous time series* when the observations are made continuously through time even when the measured variable can only take a discrete set of values (e.g., in binary processes). A time series is said to be a *discrete time series* when observations are taken only at specific times, usually equally spaced (e.g., every day at the same time).

A discrete time series can even measure continuous variables. The analyzing methods presented in the following, consider only discrete time series. Recall that every continuous time series can be transformed into a discrete one by just recording the values at equal intervals of time. This type of time series is also called a *sampled series*. Discrete time series also arise by aggregation of values over equal intervals of time[10].

Statistical theory is concerned with random samples of independent observations. In contrast, time series analysis considers successive observations that are not independent and that have an important time ordering. The fact that successive observations are dependent, future values may be predicted from past values. A *deterministic time series* assumes that a time series can be predicted exactly whereas a *stochastic time series* determines the future only partly by past observations such that exact predictions are not possible; stochastic time series use the idea the future values have a probability distribution which is conditioned by a knowledge of past values.

2.4.2 Objections of Time Series Analysis

The literature enumerates several objectives in analyzing a time series. The *description* of time series mainly includes plotting the observations against time to get a so-called *time plot*. These time plots are used to obtain simple descriptive measures of the main series properties (e.g., regular seasonal effects). *Linear systems* are used to convert an input series to an output series by a linear operation. This technique is important in the *explanation* of time series especially when observations are taken on two or more variables. Then, the variation in one time series can be used to explain the variation in another series. *Control* is another objective of time series analysis and mainly consider physical or economic systems (e.g., measure the quality of a manufacturing process).

The most important objective for this thesis is *prediction*. Given an observed time series, prediction is used to get the future values of the series. Predictions appear in sales forecasting, or in the analysis of economic and industrial time series. In general, the terms *prediction* and *forecasting* are used interchangeably. In the time series literature, there is also the differentiation between prediction to describe subjective methods and forecasting to describe objective methods. This thesis uses the term prediction in the following chapters to specify future observations.

[9]A typical example is the observation of airplane accidents.

[10]Examples of aggregated time series are monthly exports where the all exports during the whole month are accumulated.

2.4.3 Types of Variation

The literature decomposes the variation in a time series into *trend*, *seasonal variation* (or *seasonality*, or *periodicity*), and *other cyclic changes.* All other remaining types of variation are called *irregular fluctuations*.

- **Trend**: This variation is loosely defined as long-term change in the mean level. A main issue with this definition is the meaning of *long-term.* A time series of increasing or decreasing observations shows a trend.
- **Seasonality** or **Periodicity**: Time series with periodic variations show seasonal changes. Typical examples are sales figures and temperature reading where time series exhibit variation that is annual in period.
- **Cyclic Changes**: Time series with variations that do not have a fixed period but which are predictable to some extent belong to this category of cyclic changes. For example, business cycles vary from 3 or 4 years to more than 10 years.
- **Irregular Fluctuations**: Other types of variations that do not show trends, seasonal or other cyclic changes form this group of irregular fluctuations.

Trend and seasonality (or periodicity) will be in the main focus of this thesis such that prediction techniques have to consider these types of variation in the observations.

2.4.4 Prediction Techniques

Prediction or *forecasting*[11] of future values of an observed time series occurs in many areas including economics or stock control. There is a wide variety of prediction techniques available that can be grouped into three categories: *subjective*, *univariate*, and *multivariate* methods. This thesis only considers univariate methods[12] where predictions are based on present and past observations and ignores other factors (e.g., external economic influences).

Assume, there is an observed time series $x_1, x_2, \ldots, x_n$ of n values. The basic issues is to estimate future values x_{n+h} where the integer h is called the *lead time* or *horizon*. The predicted value of x_{n+h} made at time n for h time steps ahead is denoted as $\hat{x}(n,h)$ or $\hat{x}_n(h)$. Both notations emphasize the time a prediction is made and the prediction horizon.

In the following, two groups of prediction techniques will be presented in detail: *moving average* and *exponential smoothing* techniques.

2.4.4.1 Moving Average Techniques

Averaging all past observations represents the simplest prediction technique. The following equation describes simple averaging to predict $\hat{x}_n(1)$:

$$\hat{x}_n(1) = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{2.8}$$

Obviously, all past observed values have the same weight. An alternative way to summarize the past observations is to compute the mean of successive smaller sets of observed values. *Single moving average* is the simplest technique that uses the mean of the k most recent observations to predict $\hat{x}_n(1)$:

[11] Forecasting is the art of saying what will happen, and then explaining why it didn't – Anonymous
[12] Methods of this type are also called naive or projection methods

$$\hat{x}_n(1) = \frac{1}{k} \sum_{i=n-k+1}^{n} x_i = \frac{x_{n-k+1} + x_{n-k+2} + \cdots + x_n}{n} \tag{2.9}$$

The *smoothing* process is continued by advancing one period and calculating the next average of k observations, dropping the first. Two general objections about moving average techniques in general are that they cannot cope well with trends observed in the time series and assign equal weights to past observations. There exists a variation on the moving average procedure that performs better in terms of handling trend. It is called *double moving average* for a *linear trend process*. It calculates a second moving average from the original moving average, using the same value for k.

2.4.4.2 Exponential Smoothing Techniques

This group of techniques is very popular to produce a smoothed time series. Whereas in single moving average, the past observations are weighted equally, *exponential smoothing* assigns exponentially decreasing weights. In other words, recent observations are given relatively more weight in the prediction than older observations. In the case of moving average techniques, the weights assigned to the observations are the same and are equal to $1/n$ or $1/k$. In exponential smoothing, however, there are one or more *smoothing parameters* to be determined (or estimated) and these choices determine the weights assigned to the observations. Here, the focus is on the three most important exponential smoothing techniques: *single*, *double* and *triple exponential smoothing. Single exponential smoothing* (SES) is similar to the moving average technique and takes into account all past observations with exponentially decaying weights. The predicted value $\hat{x}_n(1)$ uses the recursive Equation 2.10.

$$\hat{x}_n(1) = \eta \cdot x_n + (1-\eta) \cdot \hat{x}_{n-1}(1) \tag{2.10}$$

In the beginning, single exponential smoothing uses $\hat{x}_1(1) = x_1$ such that Equation 2.10 can be used to compute predictions in a recursive manner. The parameter η is used to control the speed at which the older observations are dampened. When η is close to 1, dampening is quick and when η is close to 0, dampening is slow. The adjective *exponential* arises from the fact that the techniques uses *geometric weights* that lie on an exponential curve:

$$w_i = \eta \cdot (1-\eta)^i \tag{2.11}$$

Similar to the moving average techniques, single exponential smoothing cannot cope with trends in the observed data. *Double exponential smoothing* (DES) eliminates this weakness by taking into account trends in the observed data. This technique maintains two smoothed values L_n and T_n representing the *level* and the *trend* respectively. A predictor for the next time series value is obtained as follows:

$$\hat{x}_n(1) = L_n + T_n \tag{2.12}$$

$$L_n = \eta \cdot x_n + (1-\eta) \cdot (L_{n-1} + T_{n-1}) \tag{2.13}$$

$$T_n = \gamma \cdot (L_n - L_{n-1}) + (1-\gamma) \cdot T_{n-1} \tag{2.14}$$

At bootstrapping, DES sets $L_1 = x_1$ and $T_1 = 0$. The parameter γ is introduced to dampen the effect of trend over time, similar to η. To cope with seasonality, *triple exponential smoothing* (TES) introduces the *seasonal index* I_n besides level and trend.

Thus, three updating equations with three parameters η, γ and δ are needed. As before, the smoothing parameters are usually chosen in the range $(0, 1)$. Level, trend, and seasonal index are computed as follows where a complete season's data consists of l periods:

$$L_n = \eta \cdot (x_n/I_{n-l}) + (1-\eta) \cdot (L_{n-1} + T_{n-1}) \tag{2.15}$$

$$T_n = \gamma \cdot (L_n - L_{n-1}) + (1-\gamma) \cdot T_{n-1} \tag{2.16}$$

$$I_n = \delta \cdot (x_n/L_n) + (1-\delta) \cdot I_{n-l} \tag{2.17}$$

To initialize the triple exponential smoothing method, at least one complete season's data is needed to determine initial estimates of the seasonal indices I_{n-l}. More details can be found in the literature [Cha04]. Using the Equations 2.15, 2.16, and 2.17, triple exponential smoothing predicts the next observation $\hat{x}_n$ using the following equation:

$$\hat{x}_n(1) = (L_n + T_n) \cdot I_{t-l} \tag{2.18}$$

2.5 Distinct-Value Estimation

The task of determining the number of distinct values (*distinct-value estimation* or *DV estimation*) in a large dataset is an important question in a variety of fields in computer science, including data integration, query optimization, or network monitoring.

The exact number of distinct values can be computed by sorting the dataset and then executing a straightforward scan-and-count pass over the data; alternatively, a hash table can be constructed and used to compute the number of distinct values. Both naive approaches do not scale. Most research efforts have therefore focused on approximate methods that scale to very large datasets. These approaches either draw a random sample of the data items and use the observed frequencies of the values in the sample as a basis for estimation (e.g., [CCMN00]) or take a single pass through the data and use hashing techniques to compute an estimate using a bounded amount of memory (e.g., [Gib01]).

Here, two recent methods (*hash sketches* and *KMV synopses*) are introduced in Sections 2.5.1 and 2.5.2. Both methods support an efficient DV estimation and allow multi-set operations such as *union*, or *intersection*.

2.5.1 Hash Sketches

Hash sketches denote a statistical tool for probabilistically estimating the cardinality of a multiset S. This distinct-value estimation techniques was proposed by Flajolet and Martin in [FM85]. Hash sketches rely on the existence of a pseudo-uniform hash function $h() : S \rightarrow [0, 1, \ldots, 2^L)$, which spreads input values pseudo-uniformly over its output values. In [DF03], Durand and Flajolet further improved hash sketches (*super-LogLog counting*) by reducing the space complexity for maintaining Hash Sketches and relaxing the requirements on the statistical properties of the hash function.

2.5.1.1 Creation and DV Estimation

In their essence, the synopsis[13] works as follows. The function $\rho(y) : [0, 2^L) \rightarrow [0, L)$ is used to designate the position of the least significant 1-bit in the binary representation of y as follows:

[13]Synopsis refers to a summary or abstract. In this thesis, data structures to represent sets and multisets in a compact manner (e.g., hash sketches) are called synopses.

$$\rho(y) = \underset{k \geq 0}{min}\, bit(y, k) \neq 0, \ y > 0 \tag{2.19}$$

In Equation 2.19, $\rho(0) = L$, and $bit(y, k)$ denotes the k-th bit in the binary representation of y (bit-position 0 corresponds to the least significant bit). Estimating n, the number of distinct elements in a multiset S, proceeds as follows. For all $d \in S$, apply $\rho(h(d))$ and record the least-significant 1-bits in a bitmap vector $B[0 \ldots L]$. Since $h()$ distributes values uniformly over $[0, 2^L)$, it follows that

$$P(\rho(h(d)) = k) = 2^{-k-1} \tag{2.20}$$

With the above process, note that in the bit vector B hosting the hash sketch, $B[0]$ is expected to be set to 1 $n/2$ times, $B[1]$ $n/4$ times, etc. From this follows that the quantity $R(S) = max_{d \in S}\rho(d)$ constitutes an estimation of the value of $\log n$. The statistical error can be reduced to very small quantities by utilizing multiple bit vectors B_i, recording $\rho(h(d))$ for some item $d \in S$ to only one of the vectors B_i, producing an R_i estimate for each vector B_i, and averaging over the R_i estimates; the standard deviation of this estimation is $\frac{1.05}{\sqrt{m}}$, for m bitmap vectors [DF03].

2.5.1.2 Multiset Operations

A key property of hash sketches with great implications to the efficiency of large-scale network applications (including distributed IR) lies in the ability to combine them.

Union Operation The hash sketch of the *union* of an arbitrary number of multisets is derived from the hash sketches of each multiset by taking their bit-wise OR. Thus, given the compact synopses of a set of multisets, one can instantly estimate the number of distinct items in the union of these multisets.

More formally, if $\beta(S)$ is the set of bit positions $\rho(h(d))$ for all $d \in S$, then $\beta(S_1 \cup S_2) = \beta(S_1) \cup \beta(S_2)$. Notice that, if both original collections carry a random document, the document will conceptually be counted only once, effectively providing duplicate-insensitive (i.e. distinct item) counting for the union of the original multisets.

Intersection Operation Furthermore, hash sketches can be used to estimate the cardinality of the *intersection* (or *overlap*) of two sets. First, recall that

$$|S_A \cap S_B| = |S_A| + |S_B| - |S_A \cup S_B| \tag{2.21}$$

Second, by utilizing the union method outlined above, one can derive the hash sketch for $S_A \cup S_B$, and thus compute the cardinality of $|S_A \cap S_B|$. However, it is not possible to create the hash sketch synopsis of the intersection for future use.

The above can be generalized to more than two sets, using the *inclusion-exclusion principle* and the *sieve formula* by Poincaré and Sylvester:

$$|\bigcup_{i=1}^{n} S_i| = \sum_{k=1}^{n} (-1)^{k+1} \sum_{\substack{I \subseteq \{1,\ldots,n\},\, i \in I \\ |I|=k}} |\bigcap S_i| \tag{2.22}$$

Obviously, to compute the intersection of a huge number of hash sketches, the relative error is propagated and the distinct-value estimation is getting inaccurate. Even regarding the overlap of four multisets, the sieve formula needs a high computation complexity.

2.5.2 KMV Synopses

In [BHR+07], the *KMV synopses* and appropriate DV estimators are introduced. The main focus of KMVs is on arbitrary multiset operations including union, intersection, and differences. Their major differences when compared to hash sketches are the lower computational costs and the more accurate DV estimation. In this section, the main motivation behind the DV estimators is explained, and subsequently the KMV data structure and the basic DV estimator for them are introduced. Finally, by using the KMV synopsis and the basic estimator, the multiset operations *union*, *intersection*, and *difference* can be applied.

Assume that D points are placed randomly and uniformly on the unit interval. The expected distance between two neighboring points is $1/(D+1) \approx 1/D$, such that the expected value of U_k, the k-th smallest point, is $E[U_k] \approx k/D$. Thus $D \approx k/E[U_k]$. If U_k itself is known, a *basic estimator* for the number of points as proposed in [BYJK+02] is given in Equation 2.23:

$$\hat{D}_k^b = k/U_k \tag{2.23}$$

In the DV estimation problem, an enumeration of distinct values $v_1, v_2, \ldots, v_D$ in dataset A with domain $\Theta(A)$ is given. Using a hash function $h : \Theta(A) \mapsto 0, 1, \ldots, M$ such that the sequence $h(v_1), h(v_2), \ldots, h(v_D)$ reminds a sequence of independent and identically distributed samples from the discrete uniform distribution on $0, 1, \ldots, M$. Assuming that M is sufficiently greater than D, the sequence $U_1 = h(v_1)/M, U_2 = h(v_2)/M, \ldots, U_D = h(v_D)/M$ will approximate the realization of a sequence of samples from the continuous uniform distribution on $[0, 1]$. The requirement that M is much larger than D avoids collisions and ensures that $h(v_i) \neq h(v_j)$ for all $i \neq j$, with high probability.

2.5.2.1 Creation and DV Estimator

Using the idea of the basic estimator introduced in [BYJK+02], a KMV synopsis for a multiset S is created as described in [BHR+07]: by applying the hash function h to each value of $\Theta(S)$, the k smallest of the hashed values are recorded. This *simple* synopsis (e.g., set L_S of hashed values) is called KMV synopsis (stands for *k minimum values*).

As discussed previously, $M = O(D^2)$ is needed to avoid collisions such that each of the k hashed values requires $O(\log M) = O(\log D)$ bits. Thus, the size of the KMV synopsis is $O(k \cdot \log D)$ bits. To compute the KMV synopsis, one single scan of the dataset S is done and only a sorted list of k hashed values is necessary. KMV synopses can deal with a variety of multiset operations (union, intersection, difference). To handle multiset differences, KMVs need to be augmented with counters. These *AKMV synopses* [BHR+07] also provide the deletion of single values.

In [BHR+07], the DV estimator for KMV synopses extends the basic estimator 2.23 and uses the following computation:

$$\hat{D}_k = (k-1)/U_k \tag{2.24}$$

It is shown that this estimator is unbiased (in contrast to 2.23) and $\hat{D}_k$ is used for the multiset operations described in the following. One assumptions is that $D > k$. But, if $D \leq k$, then it is easily possible to detect this situation and return the exact value of D from the synopsis.

2.5.2.2 Multiset Operations

So far, there is an estimator for KMV synopses. Now, the focus is on multiset operations using two or more KMV synopses in combination with estimating the compound set. Here, the operations union and intersection are explained in details. To perform other multiset operations, including difference or deletion, extended KMV synopses (AKMV synopses) are needed. AKMV synopses involve adding counters to the basic synopses, in the spirit of [CG05].

In the description, it is assumed that all synopses are created using the same hash function $h : \Theta \mapsto 0, 1, \ldots, M$ where Θ denotes the data value domain appearing in the synopsis and $M = O(|\Theta|^2)$. Ordinary set operations are denoted by $\cup, \cap$ and multiset operations by $\cup_m, \cap_m$.

Union Operation Two multisets A and B with their KMV synopses L_A and L_B of size k_A and k_B, respectively, are given. The goal is to estimate the number of distinct values in the union of A and B as $D_\cup = |\Theta(A \cup_m B)|$. Here, $\Theta(S)$ denotes the set of distinct values in multiset S. Thus, $D_\cup$ can also be interpreted as $D_\cup = |\Theta(A) \cup \Theta(B)|$.

Let $L = L_A \oplus L_B$ be defined as the set including the k smallest values in $L_A \cup L_B$, where $k = min(k_A, k_B)$ and L is the KMV synopsis of size k describing $L_A \cup_m L_B$. Thus, by applying the DV estimator for KMV synopses, $D_\cup$ is estimated by following equation:

$$\hat{D}_\cup = (k-1)/U_k \tag{2.25}$$

Using the symmetric and associative operator $\oplus$, this result can be extended to multiple sets: $L = L_{A_1} \oplus L_{A_2} \oplus \cdots \oplus L_{A_n}$ estimates the number of distinct values in $A_1 \cup_m A_2 \cup_m \cdots \cup_m A_n$.

Intersection Operation As before, two multisets A and B with corresponding KMV synopses L_A and L_B of sizes k_A and k_B, respectively, are considered. The goal is to estimate $D\cap = |\Theta(A \cap_m B)| = |\Theta(A) \cap \Theta(B)|$. Set $L = L_A \oplus L_B$ with $L = h(v_1), h(v_2), \ldots, h(v_k)$, where $k = min(k_A, k_B)$. Each value v_i is an element of $\Theta(A) \cup \Theta(B)$. Also set $V_L = v_1, v_2, \ldots, v_k$ and $K_\cap = |v \in V_L : v \in \Theta(A) \cap \Theta(B)|$. Obviously, $v \in \Theta(A) \cap \Theta(B)$ if and only if $h(v) \in L_A \cap L_B$ such that $K_\cap$ can be computed from L_A and L_B alone. $K_\cap$ is utilized to estimate $D_\cap$ using the Jaccard Distance $\rho = D_\cap / D_\cup$ estimated by $\hat{\rho} = K_\cap / k$, the fraction of sampling elements in $V_L \subseteq \Theta(A \cup B)$ that belong to $\Theta(A \cap B)$. This leads to the proposed estimator:

$$\hat{D}_\cap = (K_\cap / k) \cdot (k-1)/U_k \tag{2.26}$$

Chapter 3

System Architecture and Protocols

This chapter introduces the main architecture and presents the related protocols for approximate information filtering in structured P2P networks. The architecture is named *MAPS* standing for *Minerva Approximate Publish Subscribe*, and builds upon the *Minerva* P2P Web search system to enrich one-time search with novel approximate publish/subscribe functionality. With MAPS the concept of approximate IF is introduced and the and the the first architecture to support such functionality for P2P networks is provided. Although approximate information retrieval as realized in Minerva is not the main focus of this doctoral thesis, the appropriate protocols are presented below for completeness reasons.

Section 3.1 gives an introduction to approximate information filtering, discusses the main contributions, and refers to related work in the research area of IF. Sections 3.2 and 3.3 introduce the types of services implemented in MAPS and the main protocols regulating the interactions between peers in the network. Here, the thesis also presents the appropriate retrieval protocols for one-time searching. In Section 3.4, the MAPS approach is compared to an existing exact information filtering approach for P2P networks. Section 3.5 summarizes and concludes this chapter.

3.1 Introduction

Much information of interest to humans is available today on the Web, making it extremely difficult to stay informed without sifting through enormous amounts of information. One of the most important issues is how to dig out from the *information avalanche*. In such a dynamic setting, *information filtering*, also referred to as *publish/subscribe* or *continuous querying* or *information push*, is equally important to *one-time querying*, since users are able to subscribe to information sources and be notified when documents of interest are published. This need for *content-based* push technologies is also stressed by the deployment of new tools such as *Google Alerts*[1] or the *QSR* system [YJ06]. In an IF scenario, a user posts a *subscription* (or *continuous query*) or *profile* representing his information demand (e.g., the fact that the user is interested in some sports) to the system to receive *notifications* whenever certain events of interest take place (e.g., when a paper on *distributed systems* becomes available). Information filtering and information retrieval are often referred as *two sides of the same coin* [BC92]. Although many of the underlying issues and goals are similar in retrieval and filtering, since in both cases a document needs to be matched against an information demand, the design issues, the techniques and algorithms devised to increase filtering efficiency differ significantly.

[1] *Google Alerts* (http://www.google.com/alerts) is a service offered by search engine company *Google* which notifies its users (i.e., by sending a notification email) about the latest Web and news pages of their choice.

This doctoral thesis presents how to combine approximate search and filtering functionality in a single unifying framework. Therefore, the existing Minerva system providing search capability is extended with novel approximate publish/subscribe functionality. The present thesis mainly focuses on information filtering rather than information retrieval. However, the full set of services and protocols is presented for completeness reasons. Since the focus is on IF, issues related to query routing for P2P search are only briefly discussed and the interested reader is referred to [BMWZ05, BMT+05a, MBN+06] for more details.

MAPS (*Minerva Approximate Publish Subscribe*) is a novel architecture based on the architecture of the Minerva search system to support content-based approximate information filtering in P2P environments. While most information filtering approaches taken so far have the underlying hypothesis of potentially delivering notifications from *every* information producer, MAPS relaxes this assumption by monitoring only selected sources that are likely to publish documents relevant to the user interests in the future. In MAPS, a user subscribes with a continuous query and monitors only the most interesting sources in the network. The user query is replicated to these sources and only published documents from these sources are forwarded to him. The system itself is responsible for managing the user query, discovering new potential sources and moving queries to better or more promising information sources. Since in an IF scenario the data is originally highly distributed residing on millions of sites (e.g., with people contributing to *blogs*), a P2P approach seems an ideal candidate for such a setting. However, exact pub/sub functionality has proven expensive for such distributed environments [TX03, TIK05b, AT06]. By contrast, MAPS offers a natural solution to this problem, by avoiding document granularity dissemination as the main scalability bottleneck for other approaches.

As possible application scenarios for MAPS consider the case of news filtering (but with the emphasis on information quality rather than timeliness of delivery) or blog filtering where users subscribe to new posts. Not only do these settings pose scalability challenges, but they would also incur information avalanche and cognitive overload to the subscribed users, if these were alerted for each and every new document published at any source whenever this matched a submitted continuous query. The proposed approximate IF approach ranks sources, and delivers matches only from the top-ranked providers, by utilizing novel publisher selection strategies based on a combination of *resource selection* and *behavior prediction*. Despite that the presented approach focuses on a P2P setting, notice that the MAPS architecture can also be realized in other settings, like a single server monitoring a number of distributed sources, or a farm of servers in a data center providing an alerting service. The publisher peer selection strategies will be presented in Chapter 4. In the following, the main architecture including the services and protocols is introduced.

3.1.1 Main Contributions

In the light of the above, the main contributions provided by the approximation IF approach MAPS are the following:

- A novel, network-agnostic P2P architecture, with different services and its related protocols for supporting *approximate* IF functionality in a distributed P2P environment. This is the first approach that looks into the problem of approximate IF in such a setting.
- Since, as it will be shown later, traditional *resource selection* strategies are not sufficient in this setting, this thesis devises a *novel* method to predict peer publishing behavior based on *time series analysis* of IR metrics. This technique allows to improve recall, while monitoring only a small number of publishers.

This chapter presents the services and the protocols to provide approximate information filtering. Since, the MAPS approach extends the Minerva search system, the service and protocol for one-time search are also explained for completeness reasons. Chapter 4 covers the selection strategy coming along with MAPS because publisher selection denotes the most critical task in MAPS. In Section 3.4 of this chapter, the main characteristics of MAPS are compared to an existing exact information filtering approach.

3.1.2 Previous Work on Information Filtering

Before presenting the MAPS architecture including the services and protocols in detail, previous work on information filtering is presented. The following areas are covered: *pub/sub in databases* (Section 3.1.2.1), *pub/sub in information retrieval* (Section 3.1.2.2), and *exact pub/sub in P2P networks* (Section 3.1.2.3). Since MAPS denotes the first approach for approximate IF in P2P systems, no previous work can be listed.

3.1.2.1 IF in Databases

Database research on continuous queries has its origins in the paper [TGNO92] and systems *OpenCQ* [LPT00] and *NiagaraCQ* [CDTW00]. All these papers offered centralized solutions to the problem of continuous query processing. More recently, continuous queries have been studied in depth in the context of monitoring and stream processing with various centralized [MSHR02, CF03] and distributed proposals [GL03, Ac04, JHR$^+$04]. The efforts to improve network efficiency and reduce delivery delays in content-based pub/sub systems lead to approaches like *HYPER* [ZH05], where a hybrid architecture that exploits properties of subject-based pub/sub approaches is presented.

Hermes [PB02] was one of the first proposals to use a *distributed hash table* for building a topic-based pub/sub system, while *PeerCQ* [GL03] utilized a DHT to build a content-based system for processing continuous queries. It was also one of the first approaches to assume that data is not stored in the DHT but are kept locally at external data sources. Finally, *Meghdoot* [GSAA04] utilized the CAN DHT [RFH$^+$01] to support an attribute-value data model and offered new ideas for the processing of subscriptions with range predicates and load balancing.

3.1.2.2 IF in Information Retrieval

Recently, several systems that employed an IR-based query language to support information filtering on top of structured overlay networks have been deployed. *DHTrie* [TIK05b] extended the *Chord* protocol to achieve exact information filtering functionality and applied document-granularity dissemination to achieve the recall of a centralized system at low message costs. Section 3.4 compares the DHTrie approach with MAPS to explain the architectural distinguishing features between exact and approximate information filtering in (structured) P2P networks.

In the same spirit, *LibraRing* [TIK05a] presented a framework to provide information retrieval and filtering services in two-tier digital library environments. Similarly, *pFilter* [TX03] used a hierarchical extension of the *CAN* DHT [RFH$^+$01] to store user queries and relied on multi-cast trees to notify subscribers. Again, the DHT served as a distributed index, and documents were multi-casted to reach stored subscriptions. Finally, [AT06] shows how to implement a DHT-agnostic solution to support prefix and suffix operations over string attributes in a pub/sub environment.

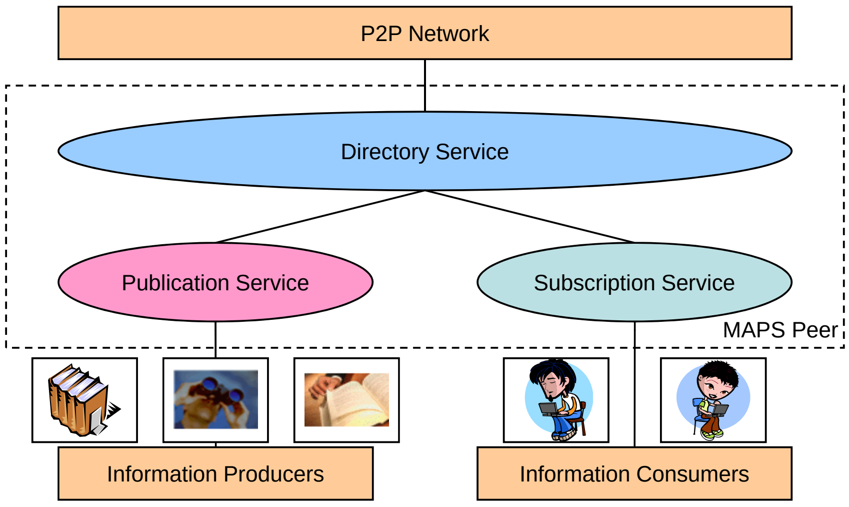


Figure 3.1: High-Level View of MAPS Service Architecture.

3.1.2.3 Exact IF in P2P Networks

All the distributed pub/sub systems described above involve some sort of *resource selection* technique to decide where to index a user query. This selection is critical, since future publications that may match this query should also be able to reach the peer storing it to trigger a notification in case of a match. Query placement, as implemented in exact information filtering approaches such as [TX03, TIK05b], is deterministic, and depends upon the keys contained in the query and the hash function provided by the DHT. These query placement protocols lead to filtering effectiveness that is exactly the same as that of a centralized system. Compared to a centralized approach, [TX03, TIK05b] exhibit scalability, fault-tolerance, and load balancing at the expense of high message traffic at publication time.

In MAPS, only the most promising peer store a user continuous query and are thus monitored by the requestor. Publications produced by each peer are matched against its local query database only since, for scalability reasons, no publication forwarding is used. Thus, in the case of approximate filtering, the recall achieved is lower than that of exact filtering, but document-granularity dissemination to the network is avoided. This improves scalability and system efficiency since, in a typical publish/subscribe setting, the rate of publications is expected to be high. In the case of exact matching, the network cost (and thus system performance) is directly dependent on this rate, whereas in MAPS approach it only triggers more local peer computations.

3.2 Services

The architecture of MAPS distinguishes the following three types of services: *directory service*, *publication service*, and *subscription service*. Peers in the MAPS network can implement any (or all) types of these services. Figure 3.1 shows an high-level overview of the service architecture including information producers, information consumers, and the P2P network.

Peers implementing the *publication service* are utilized by users to publish content to the network. This content may be locally created, e.g., from a Web server or an underlying *content management system* (CMS), or it may be gathered from the Web with a (possibly thematically focused) crawler, e.g., the *BINGO!* system [STSW02]. Additionally, other content providers such as *digital libraries* (DLs) or publishing houses (e.g., ACM, Elsevier, etc.) may also utilize publisher peers to make metadata about their resources available to the system. This setting also expects to see *observer modules* (as in [FFS+01]) for information sources that do not provide their own alerting service. These modules will query the sources for new material in a scheduled manner and inform subscribers accordingly. All this information flow will be filtered and redirected to users according to their submitted queries, by making use of different types of network peers. In this setting, information production, consumption, and flow are highly decentralized and asynchronous, making a P2P architecture a natural approach. In the following, the three service types are presented. The corresponding protocols implementing the services are shown in Section 3.3. In addition, the *one-time search service* not included in Figure 3.1 completes the whole set of services. This service is implemented by all peers in the network to perform one-time searching as in the Minerva system.

3.2.1 Directory Service

All peers participating in the MAPS network implement the *directory service*. This service provides the DHT-based routing infrastructure and is responsible for the maintenance of a distributed index storing statistics for both document and query keys. This index forms a conceptually global, but physically distributed directory, which is layered on top of a Chord-style DHT [RFH+01], and manages aggregated information about each peer's local knowledge in compact form, similarly to the directory utilized in [BMT+05a].

The DHT partitions the key space, such that every peer is responsible for the statistics of a randomized subset of keys within the global directory. To keep IR statistics up-to-date, each peer distributes per-key summaries of its *local index* along with contact information to the global directory. For efficiency reasons, these messages are piggy-backed to DHT maintenance messages and batching strategies are used. A balanced key distribution among the peers forming the directory is ensured by the DHT hash function. Additionally, to reduce the key space and also improve recall, MAPS exploits correlations among query keys and considers multi-key statistics. Chapter 5 presents two algorithms that cope with correlation awareness in MAPS, and are suited for approximate IF environments. The DHT determines which peer is responsible for collecting statistics for a specific key set. This peer maintains statistics and pointers to other peers that publish documents containing this key. The appropriate protocols are described in Section 3.3.1.

3.2.2 Subscription Service

The *subscription service* is implemented by peers that want to monitor specific information producers to get notifications for new published documents. The subscription service is critical to the recall that will be achieved at filtering time, since it is responsible for selecting the appropriate peers that will index the query. This peer selection procedure utilizes the directory service to discover and retrieve peer statistics that will guide query indexing. Once these statistics are retrieved, a ranking of the potential sources is performed and the user query is sent to the top-k ranked publishers. Only these publishers will be monitored for new publications, and since their publication behavior may not be consistent over time, query repositioning is necessary to achieve higher recall.

In the MAPS architecture, publications and subscriptions could be expressed using any appropriate IR model (e.g., Boolean, *VSM*, or *LSI*). Because, the focus is not on the data model itself, it is assumed, for simplicity, that published documents and subscriptions are sets of words, and the *Boolean model* is used to decide when a document matches an active subscription. The protocols that define the behavior of subscriber peers are explained in detail in Section 3.3.2.

3.2.3 Publication Service

The *publication service* can be used by users that want to expose their content to the MAPS network (e.g., by using a thematically focused Web crawler that locates and publishes documents on their behalf). A publisher peer utilizes the directory service to update statistics about the keys contained in the documents it publishes. Publisher peers are also responsible for storing continuous queries submitted by the users and matching them against their own publications. All queries that match a publication produce appropriate notifications to interested subscribers. Details of the publication and notification protocols are presented in Section 3.3.3.

3.2.4 One-Time Search Service

Finally, the *one-time search service* can be used by users that want to start a one-time query to receive the best matching results currently available in the P2P system. As specified in Minerva, this service utilizes the directory service to select the most promising information providers for the requested one-time query. Having selected the top-k content providers, the multi-key query is send to these peers that locally compute the matching query results. So, this service also covers the local search engine functionality of peers that return the best query results to the requestor. The last step merges all retrieved query results to one combined global result presented to the user. Details of the one-time search protocol are explained in Section 3.3.4.

3.3 Protocols

Having introduced the service architecture of MAPS, next, the main protocols – including the *directory protocol* (Section 3.3.1), the *subscription protocol* (Section 3.3.2), and the *publication* and *notification protocol* (Section 3.3.3) – are presented. Here, the thesis also presents the IR relevant *one-time search protocol* (Section 3.3.4).

3.3.1 Directory Protocol

The directory service provides the DHT-based routing infrastructure and is responsible for the maintenance of a distributed index storing statistics about document keys. This index forms a conceptually global, but physically distributed directory, which is layered on top of a Chord-style DHT [SMK⁺01], and manages aggregated information about each peer's local knowledge in compact form, similarly to [BMT⁺05a]. The DHT partitions the key space, such that every peer is responsible for the statistics of a randomized subset of keys within the directory. To keep IR statistics up-to-date, each peer distributes per-key summaries of its *local index* along with contact information to the directory. For efficiency reasons, these messages are piggy-backed to DHT maintenance messages and batching strategies are applied.

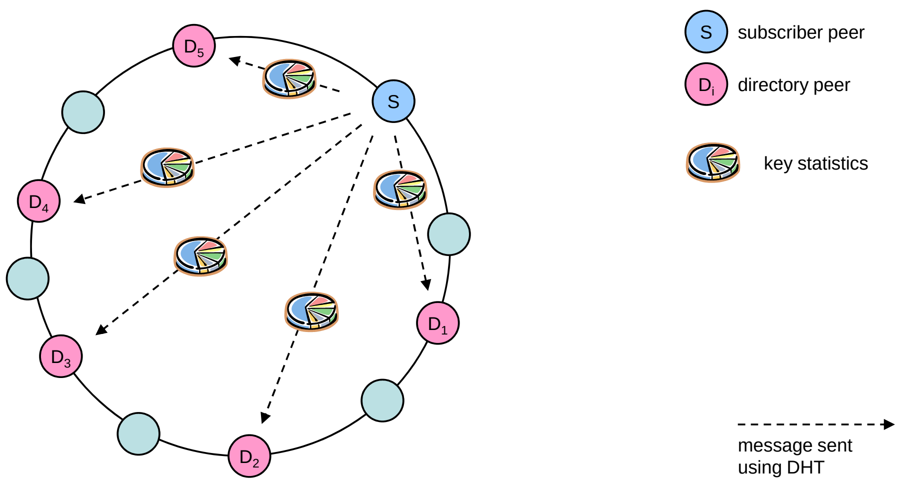


Figure 3.2: The MAPS Directory Protocol.

To facilitate message sending between peers, MAPS will use the function $send(msg, I)$ to send the message msg to the peer responsible for identifier I. Function *send()* is similar to the Chord function $lookup(I)$ [SMK+01], and costs $O(\log n)$ overlay hops for a network of n peers (see *lookup problem* in Section 2.1.1). In MAPS, every publisher peer uses Post messages to distribute per-key statistics. This information is periodically updated (e.g., every k time units or every k publications) by the peer, in order to keep the directory information as up-to-date as possible.

Publisher peer P updates the global directory as follows: Let $K = \{k_1, k_2, \ldots, k_l\}$ denote the set of all keys contained in all document publications of P occurring after the last directory update. For each key k_i, where $1 \leq i \leq l$, P computes the maximum frequency of occurrence of key t_i within the documents contained in P's collection ($tf_{k_i}^{max}$), the number of documents in the document collection of P that k_i is contained in (df_{k_i}), and the size of the document collection cs. Having collected the statistics for key t_i, P creates message Post$(id(P), ip(P), tf_{k_i}^{max}, df_{k_i}, cs, k_i)$, where $id(P)$ is the identifier of peer P and $ip(P)$ is the IP address of P. P then uses function *send()* to forward the message to the directory peer responsible for identifier $H(k_i)$ (i.e., the peer responsible for maintaining statistics for key k_i). Once a peer D receives a Post message, it stores the statistics for P in its local statistics database to keep them available on request for any peer.

Figure 3.2 illustrates the directory protocol where a peer distributes statistics (Post messages) to directory peers. Notice that the directory service does not have to use Chord or any other DHT; the MAPS architecture allows for the usage of any type of P2P network (structured or unstructured), given that the necessary information (i.e., the per-peer IR statistics) is made available through appropriate protocols to the rest of the services.

3.3.2 Subscription Protocol

The subscription protocol of MAPS is used by subscriber peers to monitor specific information producers (publisher peers) for future matching publications. It is assumed that a subscriber peer S wants to subscribe with a *multi-key* or *multi-keyword* query q of the form $k_1 k_2 \ldots k_k$ with t distinct keys.

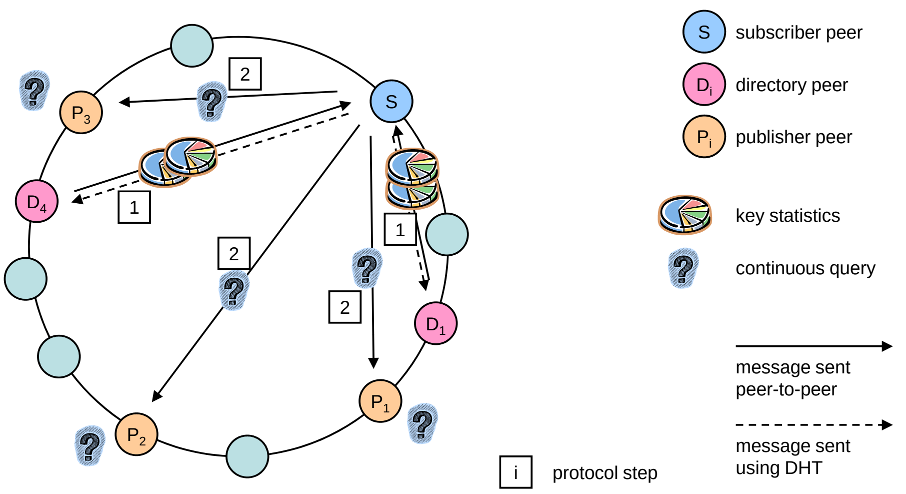


Figure 3.3: The MAPS Subscription Protocol.

To do so, S needs to determine which publisher peers in the network are promising candidates to satisfy the continuous query with appropriate documents published in the future. This source ranking can be decided once appropriate statistics about data sources are collected from the directory, and a ranking of these sources is calculated based on the peer selection strategy described in the following Chapter 4 based on *resource selection* and *behavior prediction*.

To collect statistics about data sources, S needs to contact all directory peers responsible for the query keys. Thus, for each query key k_i, S computes $H(k_i)$, which is the identifier of the peer responsible for storing statistics about other peers that publish documents containing the key k_i. Similar to the directory protocol, function $H()$ denotes the Chord hash function assigning keys to peer identifiers. Subsequently, S creates message $\text{CollectStats}(id(S), ip(S), k_i)$, and uses the function *send()* to forward the message in $O(\log n)$ hops to the peer responsible for identifier $H(k_i)$. Notice that the message contains $ip(S)$, so its recipient can directly contact S without using the DHT routing facility.

When a peer D receives a CollectStats message asking for the statistics of key k_i, it searches its local store of Post messages to retrieve the peer list L_i of all posts of the key. Subsequently, a message $\text{RetStats}(L_i, k_i)$ is created by D and sent to S using its IP found in the CollectStats message. This collection of statistics is shown in step 1 of Figure 3.3, where S contacts directory peers D_1 and D_4. Once S has collected all the peer lists L_i for the keys contained in q, it utilizes an appropriate scoring function $score(P, q)$ to compute a peer score with respect to q, for each one of the peers P contained in L_i. Based on the score calculated for each peer, a ranking of peers is determined and the highest ranked peers are candidates for storing q.

Once the peers have been ranked, S selects the highest ranked peers that will index q. Thus, only publications occurring *in those peers* will be matched against q and create appropriate notifications. Peers publishing documents relevant to q, but not indexing q, will not produce any notification, simply because they are not aware of q. Since only selected peers are monitored for publications, the peer ranking function becomes a critical component, which will determine the final recall achieved.

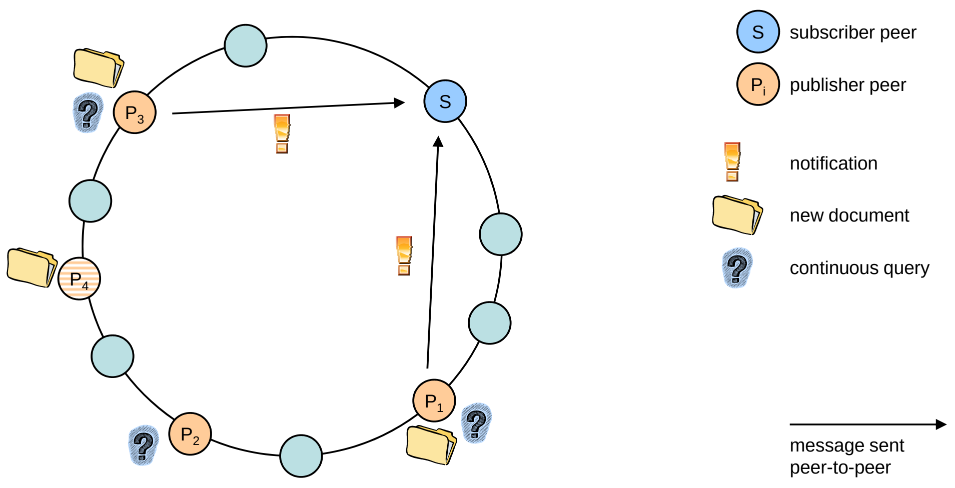


Figure 3.4: The MAPS Publication & Notification Protocol.

This scoring function can be based on standard *resource selection* approaches [LCC00] from the IR literature (e.g., CORI). However, as it will be shown in Chapter 4, these approaches alone are not sufficient in an IF setting, since they were designed for retrieval scenarios, in contrast to the IF scenario considered here, and are aimed at identifying specialized authorities. Once the peers that will store q have been determined, S uses the IP addresses associated with them to construct a message $\text{IndexQ}(id(S), ip(S), q)$. The message is forwarded to the peer that will store q. When a publisher P receives a message IndexQ containing q, it stores q using a local query indexing mechanism such as [TKD04, YGM99, TKD08]. This procedure is shown in step 2 of Figure 3.3, where S contacts publishers P_1, P_2 and P_3. Filtering and peer selection are dynamic processes, therefore periodic query repositioning, based on user-set preferences, is necessary to adapt to changes in publisher's behavior. To reposition an already indexed query q, a subscriber would re-execute the subscription protocol, to acquire new peer statistics, compute a new ranking, and appropriately modify the set of peers indexing q.

3.3.3 Publication and Notification Protocol

The publication service is employed by users that want to expose their content to the network. A publisher P utilizes the directory to update statistics about the keys contained in the documents it publishes. All queries that match a published document produce appropriate notifications to interested subscribers. According to this, the procedure followed by P at publication time is as follows.

When a document d is published by P, it is matched against P's local query database to determine which subscribers should be notified. Then, for each subscriber S, P constructs a notification message $\text{Notify}(id(P), ip(P), d)$ and sends it to S using the IP address associated with the stored query (shown in Figure 3.4). If S is not online at notification arrival, then P utilizes function *send()* to send the message through the DHT, by using the $id(S)$ also associated with q. In this way, S will receive the message from its successor upon reconnection. Notice that peers publishing documents relevant to a query q, but not storing it, will produce no notification (peer P_4 in Figure 3.4).

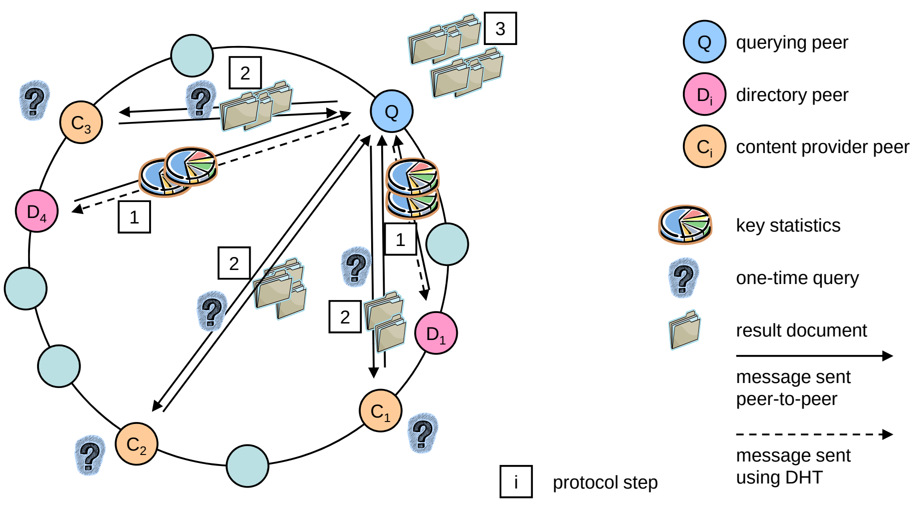


Figure 3.5: The MAPS One-Time Search Protocol.

3.3.4 One-Time Search Protocol

The one-time search protocol is similar to the subscription protocol, and is used by peers to request one-time queries to the network. It is assumed that a querying peer Q requests a *multi-key* or *multi-keyword* query q of the form $k_1 k_2 \ldots k_k$ with t distinct keys. Q needs to determine which content providers in the network are currently the most promising candidates to satisfy the query. To do resource selection, metadata about data sources is collected from the directory using the directory service and protocol. Thus, a ranking of these information sources is calculated based on resource selection techniques. In this theses, a full discussion of *query routing* in Minerva is not possible for space reasons (see [BMT$^+$05a, MBN$^+$06].

To collect statistics about data sources, Q needs to contact all directory peers responsible for the query keys. Similar to the subscription protocol, for each query key k_i, Q computes $H(k_i)$, which is the identifier of the peer responsible for storing statistics about other peers that publish documents containing the key k_i. The same $\textsc{CollectStats}(id(S), ip(S), k_i)$, is created and the function *send()* is used to forward the message in $O(\log n)$ hops to the peer responsible for identifier $H(k_i)$. A peer D receiving the message searches its local store of $\textsc{Post}$ messages to retrieve the peer list L_i of all posts of the key. Subsequently, a message $\textsc{RetStats}(L_i, k_i)$ is created by D and sent to Q using its IP found in the $\textsc{CollectStats}$ message. Figure 3.5 illustrates this first step where Q contacts directory peers D_1 and D_4. Once Q has collected all the peer lists L_i for the keys contained in q, it utilizes an appropriate scoring function $score(P, q)$ to compute a peer score with respect to q, for each one of the peers P contained in L_i. Based on the score calculated for each peer, a ranking of peers is determined and the top-k ranked peers are candidates for answering the request q. In contrast to the subscription protocol, Q sends the query q to all selected content providers contained in a message $\textsc{RequestQ}(id(S), ip(S), q)$. When a content provider C_i receives a message $\textsc{RequestQ}$ containing q, it locally executes q on its own current document collection and creates a list R_i of result documents matching the query. This result list is returned to the query initiator Q using a $\textsc{RetResults}(R_i, q)$ message. This is illustrated in step 2 of Figure 3.5.

In the third and last step of the protocol (also illustrated in Figure 3.5, the query initiator Q merges all received local result lists R_i (e.g., using appropriate merging algorithms) to provide one combined global result list to the user. Section 6.2.2 gives some more details.

3.4 Comparison to Exact Information Filtering

This Section compares the MAPS architecture for approximate information filtering presented in this chapter with an exact IF approach named *DHTrie* as introduced in [TIK05b]. This comparison considers different characteristics of both approaches [TZWK07]. Table 3.1 gives an general overview concerning them. The following sections will investigate four major characteristics concerning DHTrie and MAPS in more detail: *routing infrastructure*, *query placement*, *statistical information*, and *load balancing*. Additional experimental issues (e.g., message traffic costs) are included in Chapter 4.

	DHTrie	**MAPS**
Objective	exact pub/sub functionality (queries are indexed, all publishers are monitored)	approximate pub/sub functionality (only selected publishers are monitored)
Advantages	retrieval effectiveness (recall of a centralized system)	low network traffic, scalability, independent of publication rate
Disadvantages	dependent on publication rate, relatively high message traffic	lower recall (missing potentially interesting publications)
Routing	Chord DHT (to index queries)	Chord, Pastry DHT (to maintain statistics)
DHT Optimizations	message grouping, extra routing table	batch messaging, piggy-backing to DHT messages, key-space reduction
Query Placement	deterministic indexing (depends on query keys)	on selected peers (depends on predicted publishing behavior)
Keys Statistics	implicit key statistics (due to indexing), needed for matching	explicit key statistics (peers post and collect statistics), needed for peer ranking and matching
Load Balancing	explicit load balancing (to address imbalances), more sensitive to load imbalances	implicit load balancing (due to query placement), less sensitive to load imbalances

Table 3.1: Comparison Between DHTrie and MAPS.

3.4.1 Routing Infrastructure

A crucial design decision in both approaches is the use of the *distributed hash table* (DHT) as the underlying *routing infrastructure*. *Content-based filtering* requires an efficient object location mechanism to support expressive query languages that capture the user's specific interests. This renders *Gnutella*-style networks an inefficient solution and simple keyword functionality of standard DHTs an insufficient one.

To overcome this limitation of exact lookup, both approaches extend the DHT functionality to support richer data models and more expressive queries. However, to be able to efficiently support this functionality, changes and extensions in DHT protocols and data structures are necessary. DHTrie uses message grouping and extra routing tables to overcome inefficiencies, whereas MAPS uses batch posting of key summaries, piggy-backing of post messages to directory maintenance messages and decrease in the key space by using correlated multi-key sets to reduce network traffic (see Chapter 5).

3.4.2 Query Placement

Distributed *publish/subscribe* systems involve some sort of peer selection techniques to decide where to place a user query. This selection is critical, since future publications that may match this query should also be able to reach the peer storing it to trigger a notification in case of a match. *Query placement* in DHTrie is deterministic and it depends upon the keys contained in the query and the hash function provided by the DHT. To decide where a keyword query should be placed, all query keys are hashed and the query is forwarded to the peers responsible for these identifiers to ensure correctness at publication time. These query placement protocols lead to filtering performance that is exactly the same with that of a centralized system.

On the other hand, in MAPS only the most specialized and promising peers store a user query and are thus monitored. Publications produced by each peer are matched against its local query database only, since for scalability reasons no publication forwarding is used. In this case recall is lower than that of DHTrie, but document-granularity dissemination to the network is avoided.

3.4.3 Statistical Information

Matching incoming documents with stored subscriptions involves maintenance and estimation of important global statistics such as the document frequency of a certain key, i.e., the number of distinct documents seen in the last interval that contain a specific keyword. In DHTrie this functionality is implicit; every time a new document is published by some peer, it reaches the peers responsible for the distinct keywords contained in the document, since these are the candidate peers that may store continuous queries that will potentially match the document. These peers do all the necessary bookkeeping of the *statistical information*, and can be queried for this when other peers need to compute a similarity score.

On the other hand, MAPS explicitly addresses this issue with the maintenance of a directory. Notice that in the case of DHTrie the statistics maintenance is a byproduct of the filtering process and it is only necessary for computing similarity scores, whereas in the MAPS case it is the cornerstone of the filtering process, since both peer selection (and thus query routing) and also similarity computation utilize it.

3.4.4 Load Balancing

The effect of *load imbalances* between peers in the two approaches is different. DHTrie peers are more susceptible to load imbalances due to the nature of the subscription and publication protocol. Through the mapping of the DHT hash function, an overloaded peer or a peer with small processing capabilities may get responsible for a popular key, and thus be forced to store high volumes of user queries and process high numbers of publications. DHTrie *load balancing* mechanisms are based on load-shedding and prove efficient even for highly skewed distributions.

On the other hand, MAPS has an intrinsic load balancing mechanism to cope with imbalances. Assuming that peers are willing to offer some of their resources to the community, a peer with resources to share soon becomes a hub peer for some topic and receives more subscriptions, whereas a peer with few resources will be forced to specialize more, and thus reduce the number of users that are interested in its publications. MAPS is more sensitive to filtering load balancing (i.e., load imposed by the filtering requests that need to be processed), since a directory peer responsible for a popular key will get high numbers of statistics retrieval requests. Finally, routing load (i.e., load imposed by the messages a peer has to forward due to the overlay maintenance protocols) is similar in both systems, and it mainly depends on the DHT usage.

3.5 Discussion

This chapter presented the architecture with services and protocols of the MAPS approach for approximate publish/subscribe functionality in a structured P2P environment. While most information filtering approaches taken so far have the underlying hypothesis of potentially delivering notifications from every information producer, MAPS relaxes this assumption by monitoring only selected sources that are likely to publish documents relevant to the user interests in the future. A subscriber requesting a continuous query uses a distributed directory to collect metadata concerning publishers. Next chapter will focus on publisher peers selection approximate information filtering where a combination of *resource selection* and *behavior prediction* techniques is applied to the collected metadata such that the most promising publishers can be selected. The directory, subscription, publication and notification protocols regulate the interactions of peers in the P2P network.

For completeness, this chapter also provided the retrieval protocols for approximate one-time searching as presented in the Minerva P2P search architecture. The MAPS approach extends the Minerva system such that two functionalities (one-time and continuous searching) are available. In addition, this chapter compared the main characteristics of MAPS with an existing information filtering system called DHTrie realizing exact filtering functionality over P2P networks. This comparison highlighted the differences, advantages, and disadvantages of both architectural approaches. The next chapter will discuss the strategies to select the publisher peers that a subscriber should monitor.

Chapter 4

Publisher Peer Selection

This chapter discusses appropriate strategies to select the publisher peers that a subscriber should monitor. Notice that in MAPS peer selection is a critical task since it may lead to poor filtering effectiveness [ZTB+08].

In Section 4.1, the peer selection problem in an IF setting is introduced. Then, Section 4.2 presents the main peer selection algorithm that combines *resource selection* and *behavior prediction* to improve filtering effectiveness. The proposed prediction function of MAPS uses *time series analysis* with *double exponential smoothing* (as described in Section 2.4.4.2) to predict the future behavior of publisher peers. An extensive experimental evaluation in Section 4.3 shows the effectiveness and efficiency of the proposed algorithm. Section 4.4 develops an algorithm called *MAPS Selective Method* (MSM) to improve the prediction results without additional communication overhead, while Section 4.5 presents the benefits of the prediction improvement method MSM by analyzing several publishing scenarios. Finally, Section 4.6 discusses and summarizes the results of this chapter.

4.1 Introduction

The previous chapter introduced the architecture for approximate information retrieval and information filtering including the appropriate services and protocols. This doctoral thesis does not focus on one-time search in structured P2P networks since there is a well-known understanding concerning query routing in the Minerva search architecture. More details about Minerva query routing can be found in Chapter 6 where the prototype implementation is explained. Here, approximate publish/subscribe will be presented in detail using the MAPS approach. The most critical task in MAPS is the selection of most promising publisher peers for a requested continuous query.

The main contribution of this chapter is the presentation of the peer selection approach for MAPS by utilizing novel publisher selection strategies based on a combination of *resource selection* and *behavior prediction*. The evaluation of this approach shows that traditional resource selection strategies alone are not sufficient in this setting, and devise a *novel* method to predict peer publishing behavior based on *time series analysis* of IR metrics. This technique allows to improve recall, while monitoring only a small number of publishers. In addition, this chapter investigates an improvement algorithm called *MAPS Selective Method* to improve the prediction techniques without additional communication overhead. An extended experimental evaluation of both methods investigates the usefulness of the proposed approaches in several publishing behaviors and scenarios.

4.2 Peer Selection Strategy

The protocols presented in the previous chapter have shown that the selection of peers is the most critical part of the approximate information filtering approach MAPS. The subscriber only receives notifications about published documents from peers that store the continuous query. In other words: *a subscriber monitors a publisher with a continuous query.*

To decide which publisher peers should be monitored, the subscription protocol of Section 3.3.2 uses a *scoring function* to rank peers. In the MAPS approach, the subscriber computes a publisher peer score based on a combination of *resource selection*[1] and *behavior prediction* formulas as shown below:

$$score(P, q) = \alpha \cdot sel(P, q) + (1 - \alpha) \cdot pred(P, q) \tag{4.1}$$

In this equation, q is a continuous query, P is a certain publisher peer, and $sel(P, q)$ and $pred(P, q)$ are scoring functions based on *resource selection* and *behavior prediction* respectively that assign a score to a peer P with respect to query q. Function $score(P, q)$ is used to combine the selection and prediction scores, and decides the final score of a peer P with respect to q. The tunable parameter α affects the balance between authorities (high $sel(P, q)$ scores) and peers with potential to publish matching documents in the future (high $pred(P, q)$ scores). The value of α is from 0 to 1 where $\alpha = 1$ means that the peer score is only based on resource selection, and $\alpha = 0$ only considers behavior prediction. Based on these scores, a ranking of peers is determined and q is forwarded to the highest ranked peers. In this way, the subscriber can monitor the top-k publisher peers concerning its information demand.

Notice that this peer selection strategy is general and can also be used in centralized settings, where a server (instead of the distributed directory) maintains the necessary statistics, and mediates the interaction between publishers and subscribers.

The main contribution of this peer selection approach respect to predicting peer behavior, is to view the IR statistics provided by the (centralized or distributed) directory as *time series* and use statistical analysis tools to model publisher peer behavior. *Time series analysis* accounts for the fact that the time series observations have some sort of internal structure (e.g., *trend*, or *seasonality* etc.), and uses this observation to analyze older values and predict future ones (see Section 2.4). In the next sections, both *resource selection* (Section 4.2.1) and novel *behavior prediction* (Section 4.2.2) are presented and investigated in detail. Section 4.2.3 explains the main reasons why both components of the scoring function are needed to get a satisfying system effectiveness. An extensive discussion of resource selection for one-time search in structured P2P networks is not in the focus of this thesis. Chapter 6 includes some aspects of the Minerva one-time search approach where resource selection is adopted to overlapping P2P collections.

4.2.1 Resource Selection

In the literature, *resource selection* (or *database selection* or *collection selection*) is the task to decide to which information source a one-time query should be routed. Obviously, in a distributed system, it is too expensive to query all available information sources such that the decision is critical and strongly influences the retrieval quality (e.g., *recall* and *precision*).

[1]The term *resource selection* is also known as *database selection* or *collection selection*. *Resource selection* is throughout this thesis.

Given the large-scale data distribution of a P2P network, one of the key technical challenges is *query routing*, which is the process of efficiently selecting the most promising peers (from a potentially very large set of peers storing relevant data) for a particular information need. Query routing is very similar to resource selection that has been studied extensively in the literature in the context of *metasearch engines*[2]. There, approaches are typically designed for a small and rather static set of search engines and did not consider the challenges present in a P2P network, such as peer autonomy, network dynamics and high inter-peer latencies. The interested reader is referred to [MYL02, NF06, BMWZ05] for existing approaches to query routing.

In MAPS, the $sel(P, q)$ function returns a score for a peer P and a query q, and is calculated using standard resource selection algorithms such as $tf - idf$-based methods, *CORI* [CLC95, SJCO02], *GlOSS* [GGM95, GGMT99], the *decision-theoretic framework* (DTF) [Fuh99, NF03], or *Statistical Language Models* [SC99, PC98, XC99, ZL01, MLS99].

A P2P setting poses some new requirements to route a one-time query to an appropriate set of selected information sources. Here, the assumption of non-overlapping collections does not hold. In [BMT$^+$05a], an overlap-aware query routing approach is presented that considers the novelty of single information sources. The idea behind this approach is that a query initiator should not ask sources that provide similar content concerning a query. *Bloom filters* [Blo70] and/or *min-wise independent permutations* [BCFM00] are used to enrich the distributed directory and to compute a novelty score to combine quality and novelty measures. In addition, correlated key sets as described in [MBN$^+$06] are recognized and exploited to further improve peer selection.

Using the scoring function $sel(P, q)$, the MAPS approach identifies authorities specialized in a topic, which, as argued above, is not sufficient for the IF setting. Next, two resource selection strategies are presented in more detail including the well-known *CORI* approach and a simple approach based on IR statistics.

4.2.1.1 The CORI Approach

The *Collection Retrieval Inference Network* (CORI) [CLC95, SJCO02] reduces the task of resource selection to a document retrieval task by relying on inference networks. In CORI, a *super-document* is selected as the representative of a collection of documents, and represents the concatenation of all documents with all distinct keys of this collection. If a key appears in t distinct documents in the collection, the key appears t times in the super-document. The set of all super-documents forms a special-purpose collection that is used to identify the most promising collections for a given query. In principle, standard approaches (e.g., $tf - idf$ and *cosine measure*) could now be applied. From the viewpoint of regular document scoring and retrieval, term frequencies are replaced by document frequencies, and document frequencies by collection frequencies, i.e., the number of collections that contain a specific key.

The approach taken by CORI is implemented in the context of the *INQUERY* [CCH92] retrieval system based on *inference networks* [TC91] to compute the ranking score of a collection with respect to query q as the estimated belief that the database or information source contains useful documents. The belief is essentially the combined probability that the database contains useful documents for each query key. More specifically, the belief that the information need expressed by a query key k is satisfied by searching the collection of P is determined by the following equations:

[2]A metasearch engine is a search engine that sends user requests to several other search engines and/or databases and aggregates the results into a single list or displays them according to their source.

$$T_{P,k} = \frac{cdf_{P,k}}{cdf_{i,k} + 50 + 150 \cdot \frac{cl}{cl_{avg}}} \tag{4.2}$$

$$I_{P,k} = \frac{\frac{log(n+0.5)}{cf_k}}{log(n+1)} \tag{4.3}$$

$$sel(P,k) = 0.4 + 0.6 \cdot T_{P,k} \cdot I_{P,k} \tag{4.4}$$

Here, n denotes the total number of collections, cf_k the collection frequency, cl the number of keys in the collection P, and cl_{avg} the average number of keys in a collection. The values of some constants inserted into the formulae have been determined by empirical experiments [CLC95, SJCO02]. Note that, essentially, $sel(P,k)$ resembles the $tf - idf$ score of key k in the super-document of P. Finally, the belief that P contains useful documents with respect to query q can be computed as a sum over all query keys as in the following equation:

$$sel(P,q) = \sum_{k \in q} \frac{sel(P,k)}{|q|} \tag{4.5}$$

Compared to the original application of inference networks to document scoring, document peers are replaced by the super-documents, yielding a moderate-sized network. The frequency values are typically higher, but that does not affect the computational complexity. However, in a highly dynamic network of (cooperating) peers, the proper estimation of n, cl_{avg}, and cf is a non-trivial problem.

4.2.1.2 The MRS Approach

The *MAPS Resource Selection* approach is based on $tf - idf$ statistics directly available by the distributed directory. This simple strategy (compared to more sophisticated ones as *CORI* or *DTF*) uses the peer document frequency (df), and the maximum peer term frequency (tf^{max}) as defined before. The scoring functions $sel(P,q)$ aggregates over all query keys k as follows:

$$sel(P,q) = \sum_{k \in q} \beta \cdot \log\left(df_{P,k}\right) + (1-\beta) \cdot \log\left(tf^{max}_{P,k}\right) \tag{4.6}$$

In Equation 4.6, the value of the system parameter β can be chosen between 0 and 1 and is used to emphasize the importance of df versus tf^{max}. Experiments with various resource selection techniques in [BMWZ05] have determined that choosing $\beta = 0.5$ leads to satisfying recall results.

4.2.2 Behavior Prediction

To predict peer behavior MAPS considers time series analysis of IR statistics, thus making a rich repository of techniques from time series analysis [Cha04] applicable to this problem. These techniques predict future time series values based on past observations and differ in (i) their assumptions about the internal structure of the time series (e.g., whether *trends* and *seasonality* can be observed) and (ii) their flexibility to put emphasis on more recent observations. Details about time series analysis including different prediction techniques are presented in this thesis in Section 2.4.

Since the considered IR statistics exhibit trends, for instance, when peers successively crawl sites that belong to similar topics, or, gradually change their thematic focus, the employed time series prediction technique must be able to deal with trends properly. Further, in this scenario it is obvious to put emphasis on a peer's recent behavior and thus assign higher weight to recent observations when making predictions about its future behavior. *Moving average* techniques do not support a higher weight for more recent observations, and *single exponential smoothing* (SES) is not able to recognize trends.

For this reason, MAPS uses *double exponential smoothing* (DES) as a prediction technique, since it can both deal with trends and put emphasis on more recent observations. Section 2.4.4.2 explains double exponential smoothing and also other techniques. For completeness, notice that there is also *triple exponential smoothing* (TES) that, in addition, handles seasonality in the observed data. The MAPS setting assumes that many queries are expected to be short-lived so that no seasonality will be observed in the IR statistics time series. For an application with many long-lasting queries, one could use triple exponential smoothing, so that seasonality is taken into account. An example for seasonality could be the continuous query for *Christmas gifts*. In this case, every year before Christmas, the publication of matching documents would be recognizable.

The function $pred(P, q)$ returns a prediction score for a publisher peer P that represents the likelihood of publishing documents relevant to query q in the future. Using the double exponential smoothing (DES) prediction technique as described before, two values are predicted.

- First, for all keys k in continuous query q, $MAPS$ predicts the value for $df_{P,k}$ (denoted as $\hat{df}_{P,k}$), and uses the difference (denoted as $\delta(\hat{df}_{P,t})$[3]) between the predicted and the last value obtained from the directory to calculate the score for P. Value $\delta(\hat{df}_{P,k})$ reflects the number of relevant documents that peer P will publish in the next time-unit.

- Second, $MAPS$ predicts $\delta(\hat{cs}_P)$ as the expected difference in the collection size of peer P. This value reflects the peer's overall expected future publishing activity. If a publisher peer P does not publish any document, $\delta(\hat{cs}_P)$ will be 0.

Thus, two aspects of the peer's behavior are modeled: (i) its potential to publish relevant documents in the future (reflected by $\delta(\hat{df}_{P,k})$), and (ii) its overall expected future publishing activity (reflected by $\delta(\hat{cs})$). The time series of IR statistics that are needed as an input to the prediction mechanism are obtained using the distributed directory. The predicted behavior for peer P is quantified as follows in Equation 4.7:

$$pred(P, q) = \sum_{k \in q} \log \left(\delta(\hat{df}_{P,k}) + \log \left(\delta(\hat{cs}_P) + 1 \right) + 1 \right) \tag{4.7}$$

In the above formula, the publishing of relevant documents ($\delta(\hat{df}_{P,k})$) is more accented than the logarithmic dampened publishing rate ($\delta(\hat{cs})$). If a peer publishes no documents at all, or, to be exact, the prediction of $\delta(\hat{cs})$ and, as a consequence, the prediction of $\delta(\hat{df})$ is 0,then the $pred(P, q)$ score is also 0. The addition of 1 in the log formulas yields positive predictions and avoids $\log 0$. A simplified version of prediction function $pred(P, q)$ only considers publishing relevant documents and ignores the overall rate of peer.

[3]function $\delta()$ signifies difference.

4.2.3 Why Both Strategies Are Needed

A key component of the peer selection procedure and peer ranking function 4.1 is the *behavior prediction* mechanism introduced in this thesis. Prediction is complementary to well-known *resource selection* techniques (e.g., MRS or CORI approach). The following example using the characteristics of Table 4.1 will demonstrate the necessity in a filtering setting to combine both components of the ranking function, and shows why an approach that relies only on *resource selection* is not sufficient. Table 4.1 lists all four possible combinations describing the *past*[4] and the *future*[5].

Publisher	Resource Selection	Behavior Prediction
P_1	authority in *Sports*	publishes no *Sports*-related documents
P_2	not an authority in *Sports*	publishes *Sports*-related documents
P_3	not an authority in *Sports*	publishes no *Sports*-related documents
P_4	authority in *Sports*	publishes *Sports*-related documents

Table 4.1: Different Publisher Characteristics.

Assume a publisher peer P_1 that has specialized and become an authority in *Sports*, but publishes no relevant documents any more. Assume that P_1 will not publish any *Sports*-related documents in the future. Another publisher peer P_2 is not specialized in *Sports*, but is currently crawling a sports portal. Imagine a user who wants to stay informed about the upcoming 2008 Olympic Games, and subscribes with the continuous query *2008 Olympic Games*. If the ranking function solely relied on resource selection, peer P_1 would always be chosen to index the user's information demand, which would be wrong given that peer P_1 no longer publishes *Sports*-related documents. On the other hand, to have a high ranking score assigned, peer P_2 would have to specialize in Sports – a long procedure that is inapplicable in a IF setting which is by definition dynamic. This makes the use of a ranking and selection strategy that recognizes and predicts publishing behaviors an important component.

On the other hand, authorities are expected with high probability to publish new documents from the same topic in the future. In this case, assume another publisher peer P_3 similar to P_1 that is not publishing *Sports*-related documents at the moment. But contrary to P_1, P_3 is not a good authority for *Sports*. So, a peer selection strategy should prefer P_1 because a Sports-authority is expected to publish *Sports*-related documents in the future with a higher probability than a publisher that is specialized in another topic. For this reason, MAPS peer selection strategy combines resource selection and behavior prediction techniques. The following experiments in Section 4.3 will investigate several scenarios where a combination of both parts have to be stressed using the system parameter α in Equation 4.1. Obviously, a publisher P_4 that is an authority for *Sports* (since it stores many documents regarding *Sports*) and that also publishes new documents concerning this topic, will be the best choice for monitoring the issued continuous query.

The fact that resource selection alone is not sufficient is even more evident in the case of news items. News items have a short self-life, making them the worst candidate for slow-paced resource selection algorithms. The above example shows the need to make slow-paced selection algorithms more sensitive to the publication dynamics in the network, and MAPS employs peer behavior prediction to cope with such dynamic scenarios.

[4]The *past* describes whether a publisher is or not a good authority for a topic based on the stored data that has been published before.

[5]The *future* describes the predicted and expected publishing behavior whether a publisher will provide matching documents or not.

4.3 Experimental Evaluation

This experimental section evaluates the MAPS approach and investigates the filtering effectiveness and efficiency depending on the peer selection strategy in different publishing scenarios. Section 4.3.1 describes the experimental setup including the performance measures used in the evaluation. The experimental data is explained in Section 4.3.2 including the continuous query set. Section 4.3.3 shows experimental results for various publishing scenarios in terms of recall and benefit/cost ratio. An investigation across scenarios is included in Section 4.3.4, whereas Section 4.3.5 compares an existing exact filtering approach with MAPS. Finally, Section 4.3.6 sums up the observed experimental results of this extensive evaluation.

4.3.1 Experimental Setup

To conduct each experiment described in the next sections of this evaluation, the following five steps are executed:

1. Initially the network is set up and the underlying *distributed hash table* is created. All publisher peers join the DHT such that a structured P2P network is build-up.

2. Then, all peers in the DHT distribute metadata concerning their local collection by applying the directory protocol as described in Section 3.3.1.

3. Subsequently, subscribers utilize the protocol described in Section 3.3.2 to subscribe to selected publishers. It can be said that a publisher peer is *monitored* with a continuous query q by a subscriber, when it stores q in its local query database store, and notifies the subscriber for all publications matching q.

4. Once queries are stored, the documents are published to the network and at certain intervals (called *rounds*) queries are repositioned. A repositioning round occurs every 30 document publications per peer.

5. At the end of each round, message costs and recall for this round are calculated, and subscribers rank publishers using formula $score(P, q)$ described in Section 4.2 and reposition their queries accordingly.

To investigate the *effectiveness* and *efficiency* of the *MAPS* approach, peer publishing behavior is modeled through different publishing scenarios described in Section 4.3.3 in detail. Additionally, the two following system parameters ρ and α have to be considered in the experimental setting:

- System parameter ρ determines the percentage of publishers that a subscriber monitors after ranking them using the methods of Section 4.2. In the experimental evaluation, ρ is constant for all subscribers and different system properties for a value of ρ up to 25% are investigated. It follows that when $\rho = 100\%$ (i.e., all publishers in the network are monitored) then recall is 1.0, and the MAPS approach degenerates to exact information filtering.

- System parameter α controls the influence of *resource selection* vs. peer *behavior prediction* in the experiments. The value of α in the peer selection formula is varied from 0.0 to 1.0. A value of α close to 0.0 emphasizes peer behavior prediction, while a value close to 1.0 stresses resource selection.

To evaluate the overall filtering effectiveness of the approach the evaluation utilizes *recall*, while efficiency is measured using a *benefit/cost ratio* metric. Here, the definitions of both measures are given:

- **Recall**: The system measures *recall* by computing the *ratio* of the total number of notifications received by subscribers for all continuous queries to the total number of published documents matching subscriptions. The experiments differentiate between the recall observed in each subscription repositioning round, and the *average recall* computed over *all* rounds (i.e., for the complete experiment). Notice that measures like *precision* are not covered since the experiments assume that all documents matching a continuous query (i.e., containing all query keys) are relevant.
- **Benefit/Cost Ratio**: To evaluate the efficiency of the approach, the total number of *subscription* and *notification* messages is measured to calculate the *benefit/cost ratio* as the number of notifications per message sent. Notice that in the MAPS approach no publication messages are needed, since publications trigger only local peer computations and are not disseminated as in exact matching approaches.

As explained in the protocols of Chapter 3, the number of subscription messages depends on the number of query keys and monitored publishers. In addition, the subscription costs are proportional to the number of query *repositionings*, since for each repositioning the subscription protocol is re-executed. Finally, for each publication matching an indexed query, a notification message is created and sent to the subscriber. In the experiments of Sections 4.3.3 and 4.3.4, the message cost needed to maintain the distributed directory information is not taken into account since the main goal is to focus on the filtering protocol costs. Since directory messages are typically included in DHT maintenance messages, they can be considered as part of the underlying routing infrastructure. The directory maintenance messages are included in the cost analysis of Section 4.3.5 when MAPS is compared with DHTrie [TIK05b], an distributed exact IF approach that delivers the recall of a centralized system in a P2P setting.

4.3.2 Experimental Data

The data collection used in the experiments contains more than 2 million documents from a focused Web crawl with the focused Web crawler system *BINGO!* [STSW02]. All documents are categorized in one of ten categories: *Music*, *Finance*, *Arts*, *Sports*, *Natural Science*, *Health*, *Movies*, *Travel*, *Politics*, and *Nature*. The overall number of corpus documents is $2,052,712$. The smallest category consists of $67,374$ documents, the largest category of $325,377$ documents. In Addition, there are $36,831$ documents with unknown category (documents that are non-categorized by the focused crawler). Table 4.2 summaries the statistical overview of the experimental data collection. The number of distinct keys after *stemming* and removing *stop words* amounts to $593,876$.

In all experiments, the network consists of $1,000$ peers containing 300 documents each in their initial local collection. Each peer hosts 15% random documents (i.e., from random categories), 10% not categorized documents, and 75% documents from one single category, resulting in 100 peers specializing in each category. Using the document collection, 30 continuous queries are constructed containing two, three or four query keys as follows. Each of the query keys selected is a strong representative of a document category (i.e., a frequent key in documents of one category and infrequent in documents of the other categories). Example queries are *music instrument*, *museum modern art*, or *space model research study*. The complete list of continuous queries is shown in Table 4.3.

Category	Number of Documents
Arts	237,557
Finance	303,805
Health	213,504
Movies	82,078
Music	67,374
Nature	151,295
Natural Science	155,147
Politics	273,825
Sports	205,919
Travel	325,377
Minimum Size	67,374
Average Size	201,588
Maximum Size	325,377
Unknown Documents	36,831
Overall Collection Size	2,052,712

Table 4.2: Web Collection Statistics Grouped by Categories.

Continuous Query	
music instrument	medical risk
music instrument piano	medical risk disease
music instrument piano classic	medical risk disease heart
money invest	movie star
money invest finance	movie star video
money invest finance market	movie star video hollywood
museum modern	travel hotel
museum modern art	travel hotel offer
museum modern art design	travel hotel offer city
team sport	politics party
team sport field	politics party campaign
team sport field score	politics party campaign leader
space model	animal nature
space model research	animal nature life
space model research study	animal nature life environ

Table 4.3: Collection of 30 Continuous Queries with Two, Three, or Four Keys.

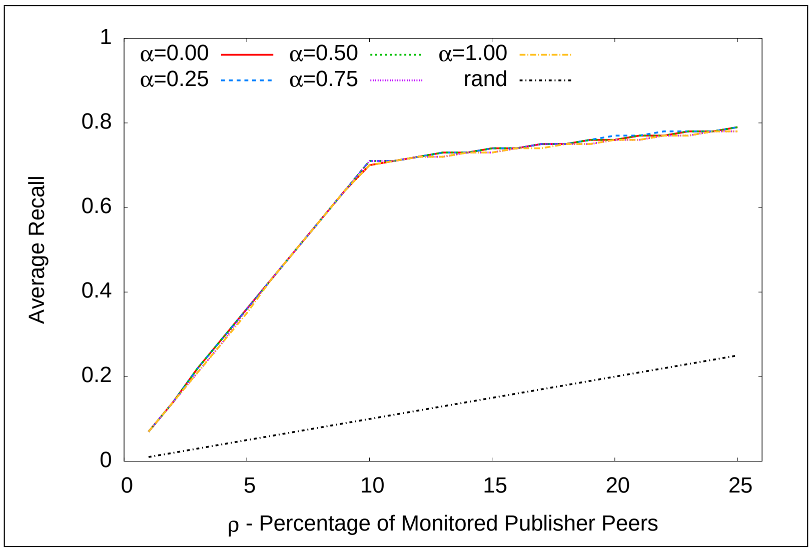


Figure 4.1: Average Recall in the *Consist* Scenario.

4.3.3 Different Publishing Scenarios

To measure MAPS's efficiency in terms of recall and message cost under various settings, five scenarios are considered representing different publishing behaviors. The overall number of published documents is constant in all scenarios ($300,000$ documents) therefore the maximum number of notifications concerning the 30 active continuous queries is also constant ($146,319$ notifications), allowing us to compare across different scenarios. The following sections shows experimental results with *average recall* and *benefit/cost ratio* for different publishing scenarios and different α and ρ values. A baseline approach (called *rand*) that implements a *random peer selection* method is included for comparison purposes.

4.3.3.1 The Consistent Publishing Scenario

The first publishing scenario *Consist* targets the performance of the approach when peers' interests remain unchanged over time. Figures 4.1 and 4.2 show that the average recall and the benefit/cost ratio do not depend on the ranking method used, and the MAPS approach presents the same performance for all values of α. This can be explained as follows. Publishers that are consistently publishing documents from one category have built up an expertise in this category and peer selection techniques are able to detect this and monitor the authorities for each topic.

Similarly, publication prediction observes this trend for consistent behavior and chooses to monitor the most specialized peers. Compared to the baseline approach of random selection (*rand*), the MAPS approach achieves up to 7 times a higher average recall (e.g., for $\rho = 10\%$). Finally, the best value for the benefit/cost ratio is when $\rho = 10\%$ that means when 10% of the publisher peers in the network store a continuous query.

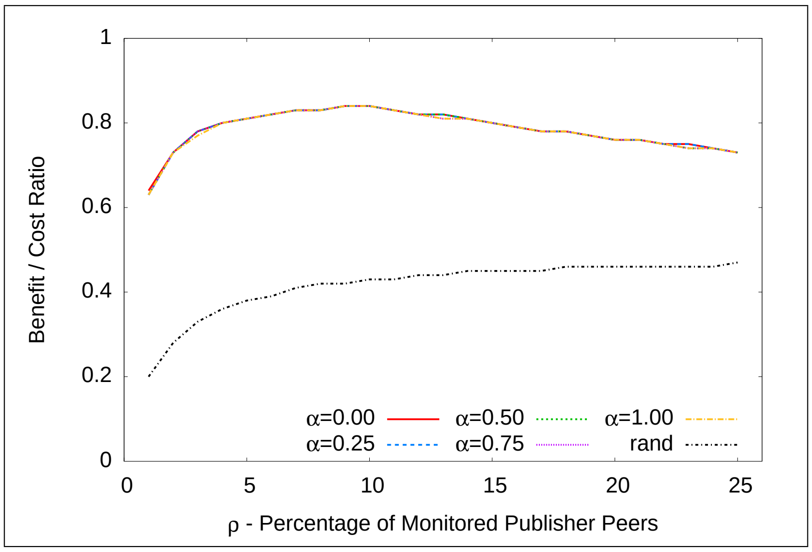


Figure 4.2: Benefit/Cost Ratio in the *Consist* Scenario.

4.3.3.2 The Half Publishing Scenario

The second scenario (called *Half*) that is observed in this evaluation series assumes that only half of the peers publish documents at all. Image a network with so-called *free-riders* that means peers only subscribing but not publishing anything. Figures 4.3 and 4.4 illustrate that the average recall is slightly higher than in the *Consist* scenario. The explanation is that MAPS peer selection recognizes that some peers do not publish documents at all. The benefit/cost ratio is also higher than in in previous scenario. It reaches the highest benefit/cost ratio for $\rho = 5\%$. As in the previous scenario, MAPS outperforms the random peer selection approach *rand* for all parameter settings.

4.3.3.3 The Category Change Scenario

Since users may publish documents from different topics, the evaluation uses the scenario *CatChg* to simulate the changes in a publisher's content. In the *CatChg* scenario, a peer initially publishes documents from one category, and later on switches to a different category at some point in time. Figures 4.5 and 4.6 illustrate the performance of the approach in this scenario for different values for α and ρ. The most important observation from these figures is the performance of the prediction method in comparison to resource selection. In some cases (e.g., when $\rho = 10\%$) not only publication prediction achieves more that 6 times better average recall than resource selection, but also resource selection is only marginally better than *rand* (e.g., when monitoring 15% of publishers). So, this scenario affirms the example of Section 4.2.3 why resource selection is not sufficient for filtering scenarios.

In general, both average recall and benefit cost/ratio improve as α reaches 0.0 and prediction is stressed. This abrupt changes in the publishers' content cannot be captured by any resource selection method, which favors topic authorities. On the other hand, publication prediction detects the publishers' topic change from observed changes in the IR statistics and adapts the scoring function to monitor peers publishing documents relevant to subscribed continuous queries.

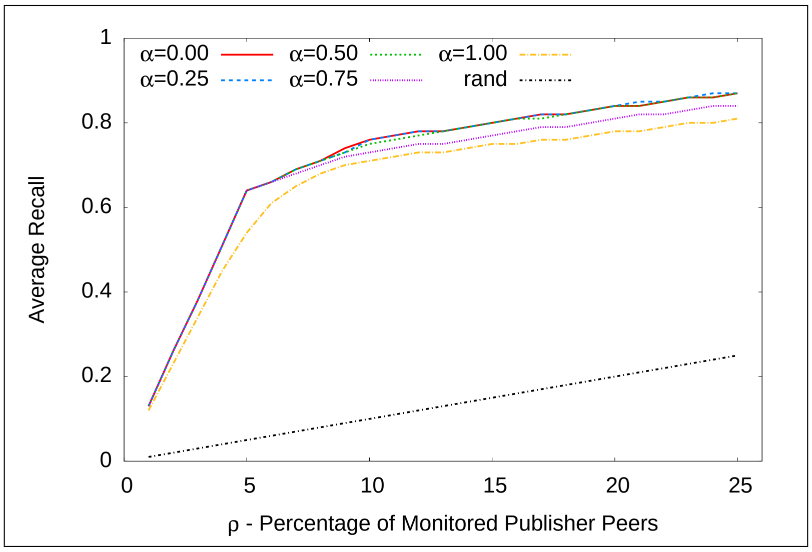


Figure 4.3: Average Recall in the *Half* Scenario.

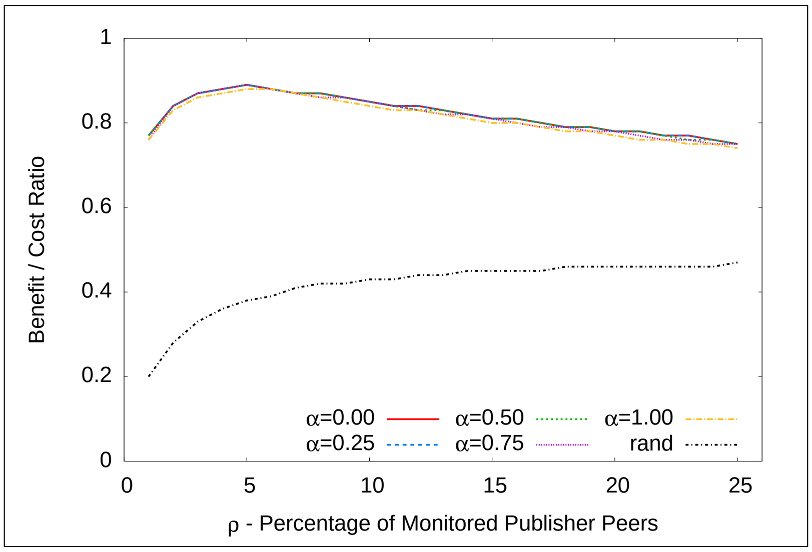


Figure 4.4: Benefit/Cost Ratio in the *Half* Scenario.

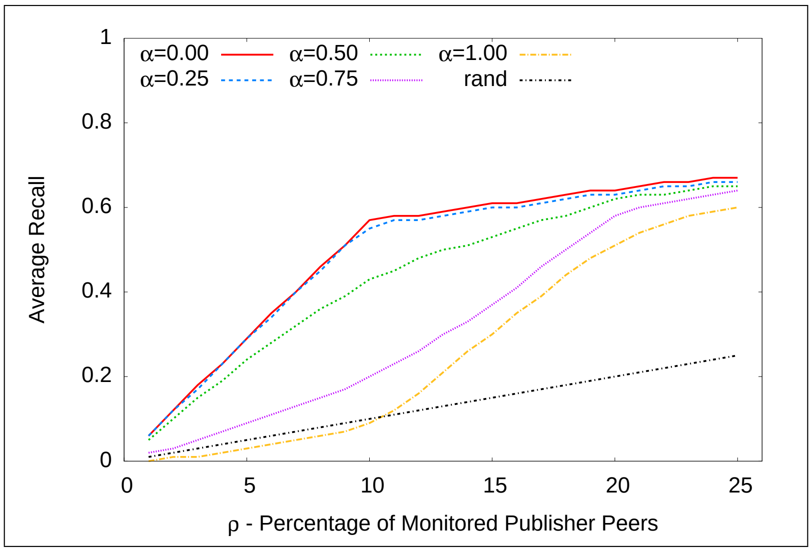


Figure 4.5: Average Recall in the *CatChg* Scenario.

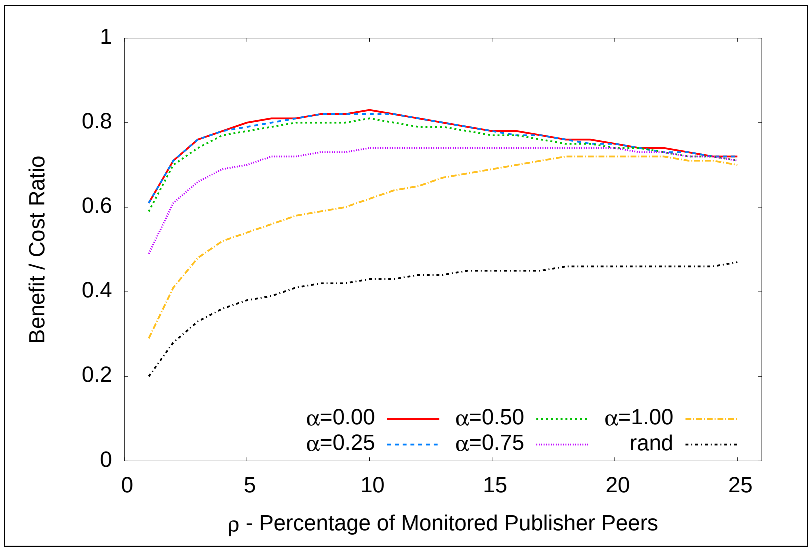


Figure 4.6: Benefit/Cost Ratio in the *CatChg* Scenario.

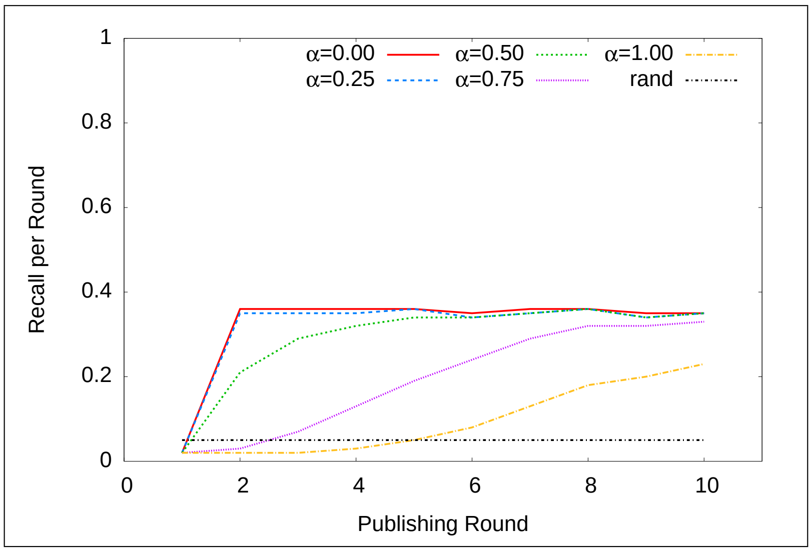


Figure 4.7: Recall per Publishing Round for $\rho = 5\%$ in the *CatChg* Scenario.

The experiments shown in Figures 4.7 and 4.8 are introduced to demonstrate the retrieval effectiveness of MAPS in each of the 10 query repositioning rounds for the *CatChg* scenario, when α is variable and for $\rho = 5\%$ and $\rho = 10\%$[6]. In round one, all combinations of publication prediction and resource selection provide the same performance because no time series values are available yet; thus only the resource selection method is used to rank peers. As new statistics become available in later rounds, the prediction method improves recall. The more emphasis is put on the prediction part of the ranking, the faster the learning process evolves, as is clearly shown by the recall increase. A comparison of the two figures reveals that the learning process of the scoring function remains unaffected by ρ, and that prediction manages to adapt quickly in topic shifts, by needing only two repositioning rounds to reach the maximum recall.

4.3.3.4 The Publishing Breaks Scenario

The *Break* scenario models the behavior of peers as they log in and out of the network. It is assumed that some publishers are active and publish documents for some rounds, and then log out of the network, publishing no documents any more. This procedure is continued in intervals, modeling, e.g., a user using the publication service at home, and switching it off every day in the office. The ranking mechanism should recognize and adapt to these inactivity periods, and distinguish between peers not publishing documents any more and peers making temporary pauses. Figures 4.9 and 4.10 demonstrate that both average recall and benefit/cost ratio improve when resource selection is emphasized (i.e., when α is close to 1.0), since pauses in the publishing mislead the prediction formula to foresee that, in the future, no relevant publications will occur. For this reason, peers with inactivity periods are ranked lower resulting in miss of relevant documents. On the other hand, resource selection accommodates less dynamics, so temporary breaks remain undetected and the topic authorities continue to be monitored since the ranking procedure is not affected.

[6]Monitoring 5% or 10% of the network seems to be a realistic size.

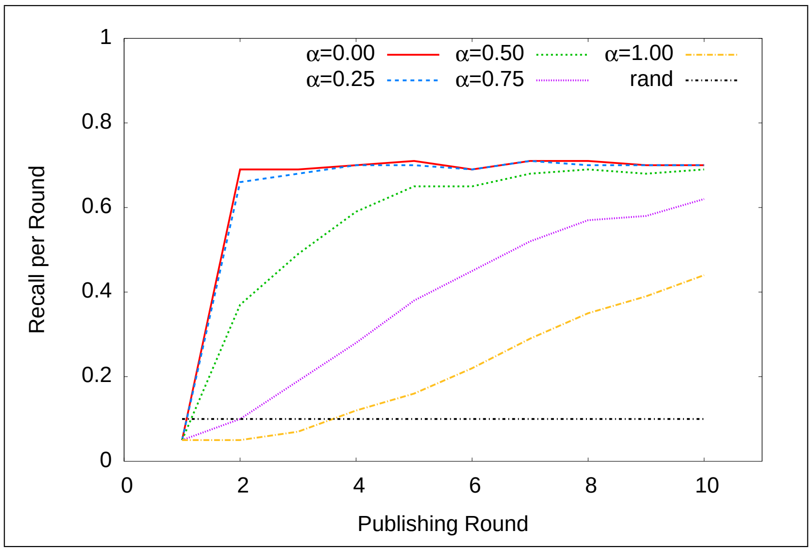


Figure 4.8: Recall per Publishing Round for $\rho = 10\%$ in the *CatChg* Scenario.

Consequently, selecting a ranking method that favors prediction leads to poor recall and low benefit/cost ratio, that are comparable to those of *rand*. This scenario also fits to the example presented in Section 4.2.3 where the use of resource selection can be argued by the higher probability to publish *related* content.

4.3.3.5 The Temporary Changes Scenario

The last scenario investigated (*TmpChg*), targets temporary changes in a peer's published content. This scenario models users utilizing the service e.g., for both their work and hobbies, or users that temporarily change their publishing topic due to an unexpected or highly interesting event (earthquake, world cup finals, etc.). Here, a publisher makes available documents about one topic for a number of rounds, and then temporarily publishes documents about a different topic. In the next rounds, the publisher reverts between publishing documents out of these categories, to stress the behavior of peers being interested in a topic but occasionally publish documents covering other topics.

In this scenario, MAPS presents the highest average recall values when equally utilizing *resource selection* and *prediction methods* ($\alpha = 0.5$), as suggested by Figures 4.11 and 4.12. This happens because *TmpChg* can be considered a scenario lying between an abrupt category change (*CatChg* scenario) and publishing documents about a specific topic with small breaks (*Break* scenario). Thus, the combination of publication prediction and resource selection used by subscribers, aids in identifying these publication patterns in publisher's behaviors and thus selecting the peers publishing more relevant documents.

Finally, an interesting observation emerging from these figures is that almost all combinations of ranking methods perform similarly both in terms of average recall and benefit/cost ratio. This is due to the effectiveness of the ranking methods, that cause the dampening of the subscription messages by the high number of notification messages created. Compared to the baseline random peer selection *rand*, all methods show an increase of as much as 600% for average recall and 200% for benefit/cost ratio. This improvement shows the potentials of the MAPS filtering approach.

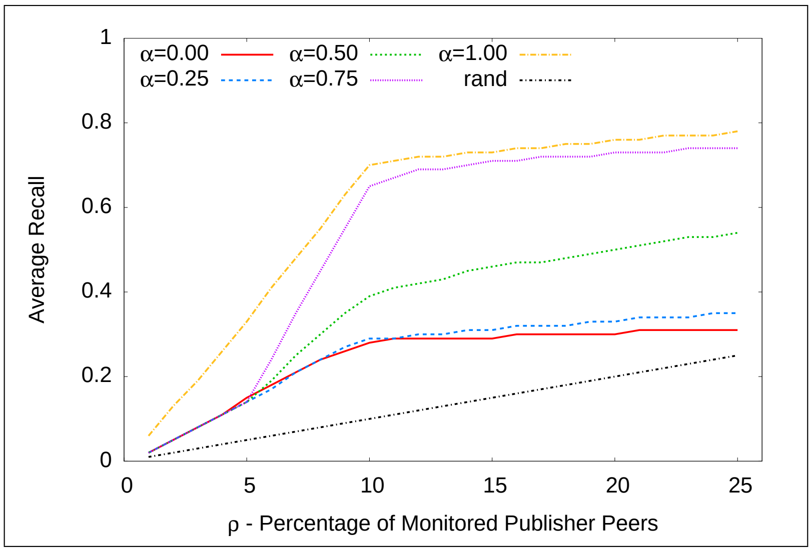


Figure 4.9: Average Recall in the *Break* Scenario.

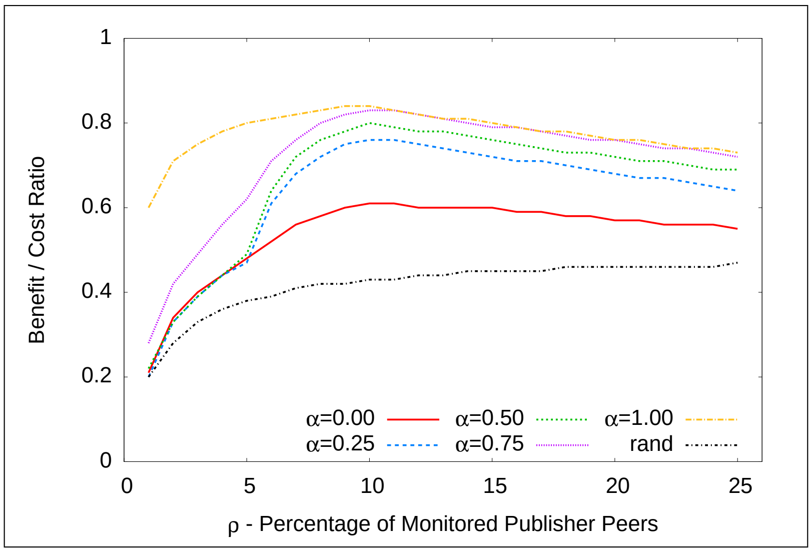


Figure 4.10: Benefit/Cost Ratio in the *Break* Scenario.

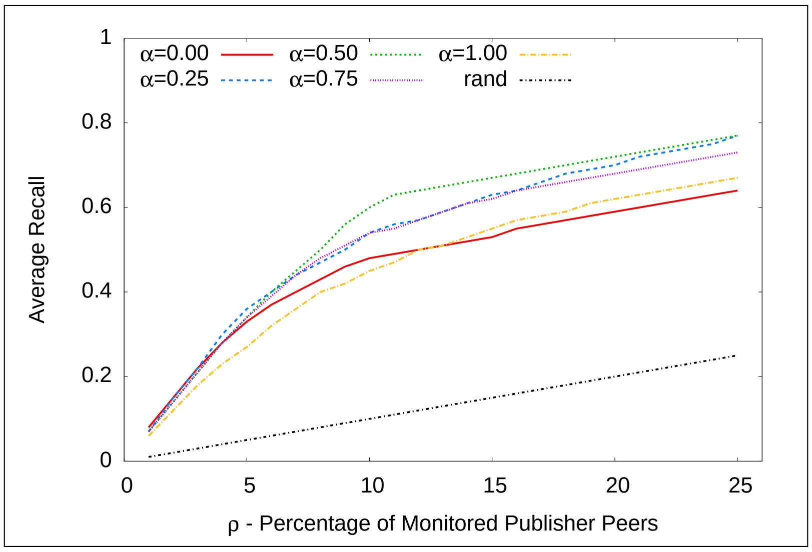


Figure 4.11: Average Recall in the *TmpChg* Scenario.

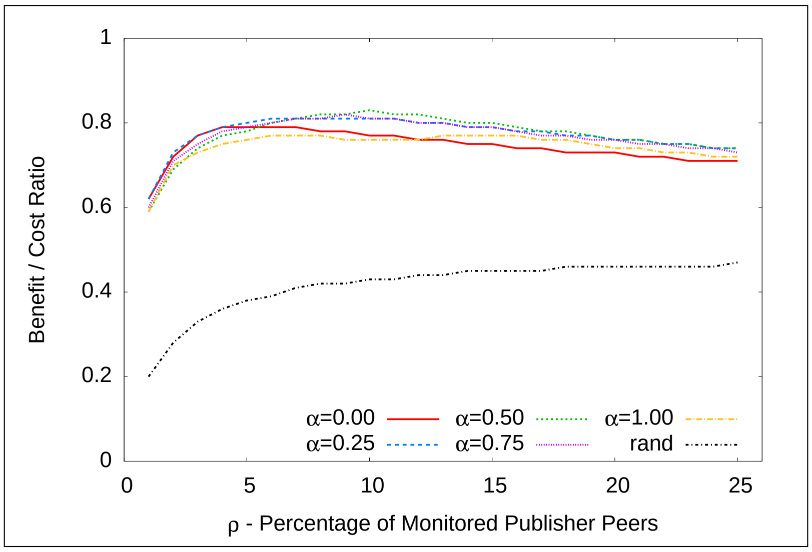


Figure 4.12: Benefit/Cost Ratio in the *TmpChg* Scenario.

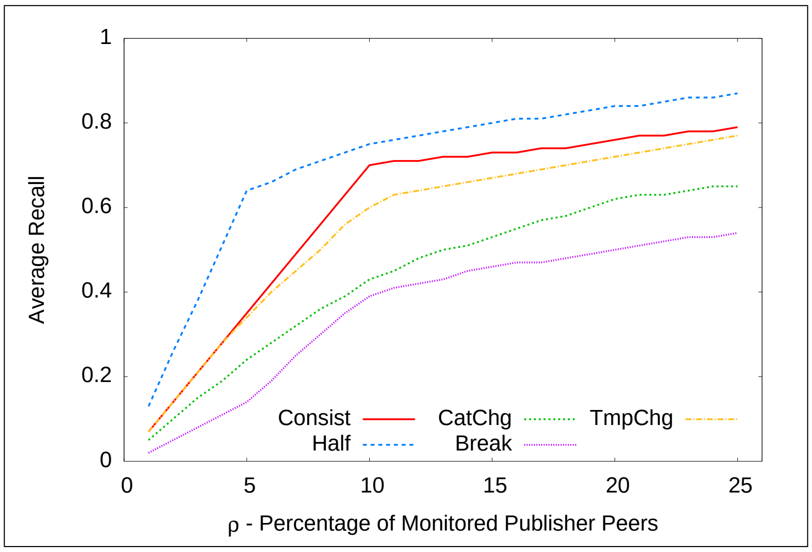


Figure 4.13: Average Recall for $\alpha = 0.5$ in Different Scenarios.

4.3.4 Comparison Across Scenarios

In this section, the experimental viewpoint is changed. The evaluation selects some baseline values for α and ρ, and compare the average recall and the benefit/cost *across scenarios* to get some more insights. Reasonable baseline values for the experiments in this section are $\alpha = 0.5$ and $\rho = 10\%$.

4.3.4.1 Average Recall Analysis

Figures 4.13 and 4.14 illustrate the *average recall* values achieved for the various publishing scenarios. When $\alpha = 0.5$ and ρ value increases up to 25% of monitored publishers (Figure 4.13), it is shown that *Consist* achieves the highest average recall, since, as explained in Section 4.3.3.1, it is not affected by the choice of α. The rest of the scenarios achieve lower average recall with the *TmpChg* scenario being the most promising. For the rest of the scenarios the choice of $\alpha = 0.5$ is a compromise that leads to satisfactory average recall.

When ρ is set to 10% and the emphasis of the ranking method moves from behavior prediction ($\alpha = 0.0$) to resource selection ($\alpha = 1.0$), the average recall remains relatively unaffected for *Consist* and *Half* publication scenarios (Figure 4.14). In contrast, *CatChg* and *Break* are influenced by the applied ranking method, and demonstrate a significant change in their behavior and average recall achieved. The *TmpChg* scenario reaches the highest average recall levels when both publication prediction and resource selection are equally weighted with $\alpha = 0.5$.

4.3.4.2 Message Costs Analysis

The *benefit/cost ratio* for the different publishing scenarios is shown in Figures 4.15 and 4.16. Here, the value of ρ increases to demonstrate the benefit/cost ratio for a constant $\alpha = 0.5$ (Figure 4.15) and the dependency of the benefit/cost ratio parameter on the ranking method is illustrated as a function of α for a constant ρ of 10% (Figure 4.16).

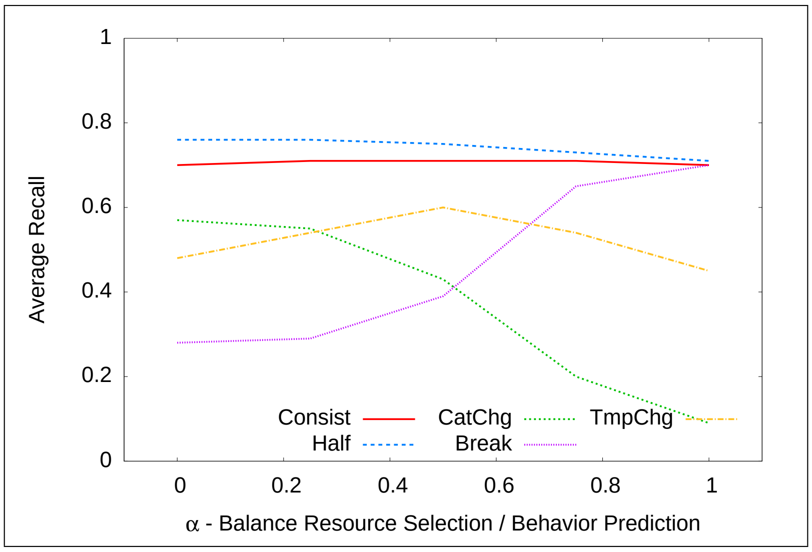


Figure 4.14: Average Recall for $\rho = 10\%$ in Different Scenarios.

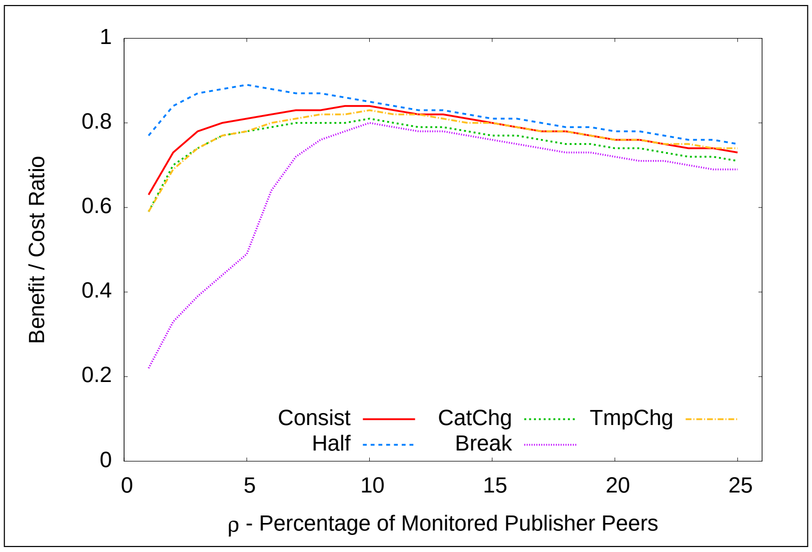


Figure 4.15: Benefit/Cost Ratio for $\alpha = 0.5$ in Different Scenarios.

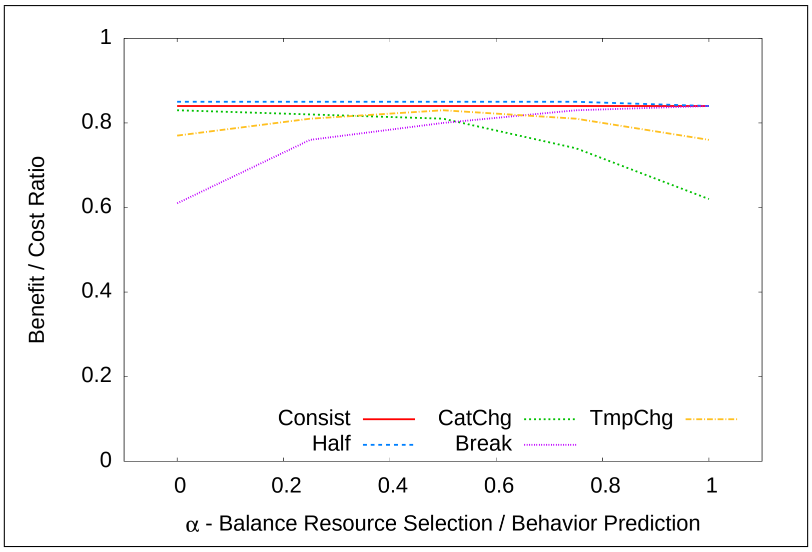


Figure 4.16: Benefit/Cost Ratio for $\rho = 10\%$ in Different Scenarios.

The most important observation from these experiments is that independently of the ranking method used, in all scenarios, the highest value for the benefit/cost ratio is achieved when monitoring 10% of the publisher peers (*Half* is an exception with highest ratio for 5%). Obviously, this observations is partly caused by the construction of the publishers since there are 10% of the peers specialized in each category[7]. At this value, the MAPS approach needs around 1.2 messages per notification generated (since the number of notifications/message is around 0.8 as shown in the graphs). Obviously, the best possible benefit/cost ratio is 1.0, since at least one message (the notification message itself) is needed at publication time to inform a subscriber about a matching document. This means that an average of 0.2 extra subscription messages per notification sent are generated in the experimental evaluation.

It can be observed that in the *Consist* scenario, a change in the ranking method has no effect on the value of the benefit/cost ratio. The same holds for the *Half* scenario. Nevertheless, the *Break* and *CatChg* scenarios perform differently such that the benefit/cost ratio increases for the case of resource selection and publication prediction respectively. The *TmpChg* scenario differs from all other scenarios because, for the same reason as in Section 4.3.4.1, the highest benefit/cost ratio is achieved when combining both resource selection as peer prediction scores.

4.3.5 Comparison to Exact Information Filtering

In the last set of experiments, the evaluation studies the message cost imposed by the MAPS approach on the network as a function of the average recall achieved. The results are shown in Figure 4.17. In addition, this a short comparison to an existing exact filtering system called DHTrie [TIK05b] is presented and the results are shown in Figure 4.18. Notice that *DHTrie* is an exact filtering approach that delivers notifications to all matching publications by disseminating documents in the network.

[7]In the *Half* scenario only half of all peers publish documents.

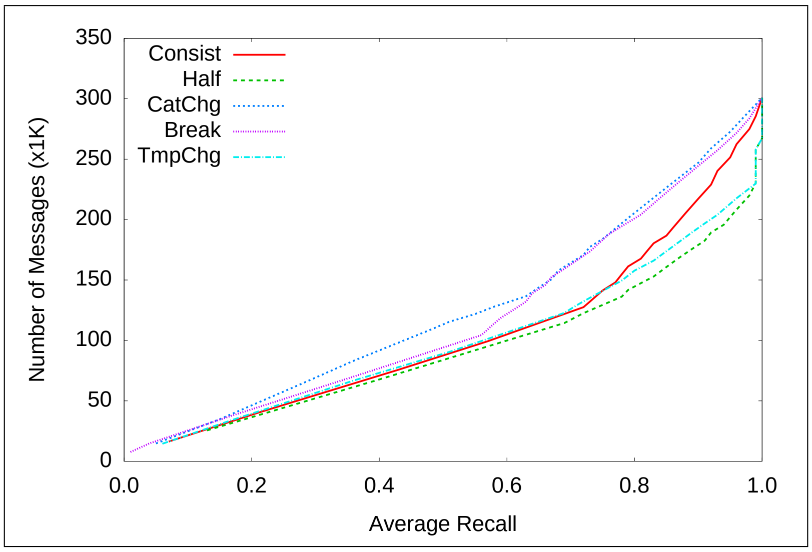


Figure 4.17: Number of Messages and Average Recall Level.

Figure 4.17 presents the costs in filtering messages needed to achieve a certain level of average recall for $\alpha = 0.5$. As can be seen, to achieve average recall equal to 1.0 (i.e., as in an exact filtering approach), roughly $300,000$ filtering messages are needed independently of the scenario in question. Additionally, for all scenarios, the MAPS approach achieves an average recall between 65% - 85% by using only 50% (i.e., about $150K$ messages) of the total messages needed to achieve a recall of 1.0.

The evaluation has not considered the directory maintenance cost so far, since the main focus was on recall and filtering cost imposed on the network. To compare the MAPS approach with exact IF, all types of network costs have to be included. Thus, the evaluation used an implementation of the DHTrie protocols of [TIK05b] and compared the network costs between approximate and exact IF.

Figure 4.18 shows the total message cost (including directory maintenance and filtering messages) as a function of the total number of documents published in the system for both approaches. As shown, approximate filtering costs are independent of the number of publications, since publications trigger only local computations. On the other hand, *DHTrie* (and also all other exact filtering approaches) are sensitive to this parameter since documents have to be disseminated to the network to meet stored queries. Thus, approximate filtering improves network costs up to a factor of 8, at the expense of lower recall.

In [BMW06], several strategies to decrease directory maintenance costs in a retrieval setting are presented. The main idea behind these strategies, is that if the statistics of the low-ranked peers are not regularly posted to the directory, peer selection quality does not decrease, while directory maintenance cost are significantly reduced. In the evaluation this technique is exploited and two approaches to threshold the posted statistics in the IF setting are utilized: an *absolute* and a *relative threshold strategy*. In *absolute thresholding*, peers do not take into account the size of their document collection for setting the threshold, whereas *relative thresholding* adapts to peer collection size to allow for a more flexible pruning. The proposed strategies manage to lower message costs by a factor of 11 compared to the full directory maintenance costs while having a recall loss of less than 4%.

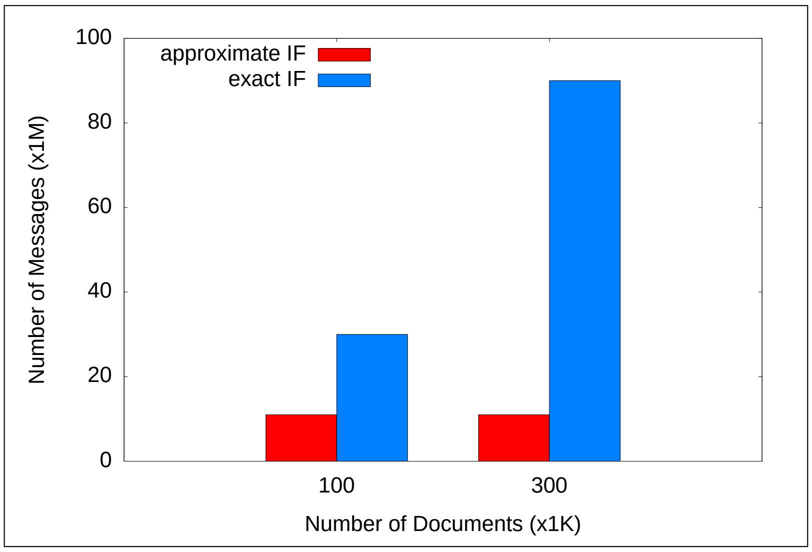


Figure 4.18: Comparison Exact vs. Approximate Information Filtering.

4.3.6 Summing Up

The evaluation has shown the tradeoffs of the MAPS approach with respect to retrieval effectiveness and network load, and also in comparison with exact IF approaches. In MAPS, most of the effort is spent at query indexing time, to minimize network traffic at publication time. It presents minimal message costs since only 0.2 messages per matching document are created. Additionally, by exploiting peer specialization, it can achieve recall as high as 80% by monitoring only 8% of the publishers. To maximize system performance, a judicious choice of system parameter values should take into account the specifics of each application. In scenarios with high dynamics, publication prediction should be stressed, whereas, in scenarios with high peer specialization and less dynamics, resource selection should be favored. Finally, a comparison to exact IF showed that MAPS trades 10% - 20% of recall to reduce message traffic by a factor of 8.

4.4 Improving Prediction

This section investigates strategies to improve behavior prediction in the presented approximate filtering approach. In *MAPS*, the *double exponential smoothing* (DES) technique is used to predict the future publishing behavior of publisher peers, and this prediction is used to calculate peer scores that rank publisher peers according to the likelihood to publish in the future documents of interest. Besides this property, *DES* assigns exponentially decreasing weights to older values and is able to recognize *trends* in a time series of values. To achieve this, DES utilizes two parameters, η and γ, that are used to tune the prediction formula to follow a more aggressive or passive prediction of values (see Section 2.4.4.2). In this section, the most appropriate parameter choices for different value behaviors are investigated. The claim is that one *global* combination for η and γ is not sufficient to recognize individual publishing behaviors of different peers, since it may lead to significantly lower recall.

For this reason, the *MAPS Selective Method* as described in [ZTB+07] is introduced to compute the best parameter setting per peer. This results in the reduction of prediction errors and significant recall improvements since individual publishers may present various behaviors. The method is scalable, since it does not incur additional communication costs between peers, but rather utilizes information that is locally available at each peer. In the extensive experimental evaluation the MAPS Selective Method is compared against two opponents:

- An *oracle* that always predicts the monitored values accurately. As a consequence, this opponent describes the best possible recall achievements for MAPS using behavior prediction techniques.
- An opponent selecting parameter combinations that require a *global view* of the network, which is highly inefficient for large-scale approaches since additional communication overhead is incurred.

The *MSM* approach manages to perform almost as good as the oracle and centralized opponents, but without incurring any extra communication cost since only local communication has to be applied.

Based on the equations of *double exponential smoothing* presented in 2.4.4.2, different publishing behaviors are analyzed in Section 4.4.1. In 4.4.2, the *MAPS Selective Method* is introduced to automatically adapt the double exponential smoothing technique to the observed data series. MSM uses an algorithm that does not need any additional communication cost. Next, alternative approaches are discussed in 4.4.3. The following Section 4.5 presents an extensive experimental evaluation of the MAPS Selective Method using a Web data collection and publisher peers with different behaviors. Average recall and prediction errors are used as experimental measures.

4.4.1 Analyzing Different Behaviors

To provide a better understanding of the exponential smoothing techniques and especially *DES*, there is an investigation how double exponential smoothing is able to predict the correct future values. For this, eight different data series are assumed, each simulating a different publishing behavior. Table 4.4 shows all behaviors with corresponding observations. All values range from 0 to 600, and the series length is 10. Table 4.4 also shows how the i-th value of the series is computed. The behaviors *LinDec* and *LinInc* show a linear trend since, here, a linear time series is described. All six other behaviors (*LogInc*, *LogDec*, *QuadInc*, *QuadDec*, *ExpInc*, and *ExpDec*) clearly show a non-linear trend in their data value. For simplification reasons, no data series shows *periodicity* (or *seasonality*) thus *triple exponential smoothing* is not necessary.

Subsequently, the influence of the two double exponential smoothing parameters η and γ is investigated by looking at all possible combinations from 0.0 to 1.0 in steps of 0.1 for both parameters. This way, there are overall 121 different parameter combinations for η and γ. For each combination, the first four data values of the different behaviors are used as bootstrapping values, and the average absolute prediction error is computed per round between the real data observation and the predicted observation using double exponential smoothing technique. Figure 4.19 shows for each behavior the corresponding absolute prediction errors (y-axis) when varying the average prediction *Avg* value per parameter combination (x-axis) in descending order. Consider that parameter combinations are not in numerical order for η or γ.

Behavior	Properties
LogInc	$log(i) * (600/\log 10)$ $0, 180, 286, ..., 541, 572, 600$
LogDec	$log(10 - i + 1) * (600/\log 10)$ $600, 572, 541, ..., 286, 180, 0$
LinInc	$(600/10) * i$ $60, 120, 180, ..., 480, 540, 600$
LinDec	$(600/10) * (10 - i + 1)$ $600, 540, 480, ..., 180, 120, 60$
QuadInc	$(i^2) * (6)$ $6, 24, 54, ..., 384, 486, 600$
QuadDec	$((10 - i + 1)^2) * (6)$ $600, 486, 384, ..., 54, 24, 6$
ExpInc	$600/(10 - i + 1)$ $60, 66, 75, ..., 200, 300, 600$
ExpDec	$600/i$ $600, 300, 200, ..., 75, 66, 60$

Table 4.4: Data Series Simulating Different Publishing Behaviors.

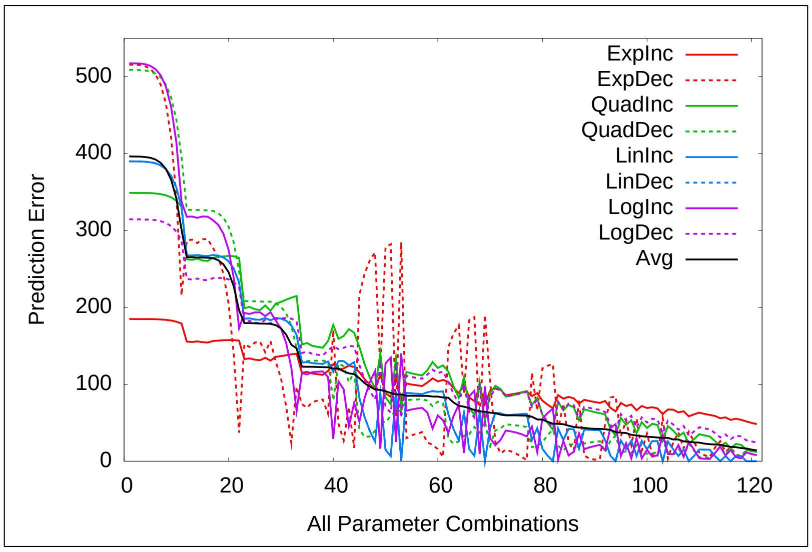


Figure 4.19: Prediction Errors with DES for Different Behaviors.

As a result, Figure 4.19 illustrates that there is a high variation of prediction errors depending on the choice of the parameter combination of η and γ. Although different value pairs may lead to small prediction error for double exponential smoothing, notice that there is no *optimal* parameter combination. The best combination for one behavior does not necessarily result in satisfying predictions for another one. The conclusion of this observation is that a *global choice* for η and γ cannot be used to predict *all* peer behaviors for *all* active continuous queries. This is reasonable since some behaviors show stronger or less stronger trends that have to be addressed with DES. As a result, *MAPS* has to adapt the parameter combination for η and γ to the observed value behavior. In the next section, the *MAPS Selective Method* is presented as a novel method for capturing parameter combinations that will result in high recall at no extra communication cost.

4.4.2 The MAPS Selective Method

The conclusion of the previous investigation is that there exists no single setting for parameters η and γ to effectively model all different publishing behaviors. Therefore, the *MAPS Selective Method* is introduced to adapt the parameter setting to the given scenario. The approach works as follows.

Let the values $x_1, \ldots, x_{n-1}$ denote the observed time series values and let $\hat{x}_n$ be the predicted value. The Selective Method uses the values $x_1, \ldots, x_{n-2}$ to predict the already known last observed value x_{n-1}. Let $\hat{x}_{n-1,\eta,\gamma}$ denote the predicted value for all combinations of η and γ. Now, the parameter combination with the smallest error concerning the real observed value of x_{n-1} is selected. If there are more than one combination with smallest error, the one with the lowest distance to $\eta = 0.5$ and $\gamma = 0.5$ is picked. The selected parameters are used to predict the next future value $\hat{x}_n$.

The *MSM* algorithm uses the observed data series $x_1, \ldots, x_{n-1}$ and the function $DES_{\eta,\gamma}$ as input to predict the next value. The algorithm outputs the selected parameter values for η and γ, as well as the prediction value $\hat{x}_n$ as the result of $DES_{\eta,\gamma}$. Notice that the algorithm is simplified such that the case that several parameter combinations have the smallest error is not considered. Moreover, the observed time series needs at least two values such that the MSM algorithm can be applied.

The Selective Method means that the most appropriate parameter setting concerning the last observed value is always used. In this way, double exponential smoothing is adapted to the given data series. Obviously, at least four observed values are needed to apply this method because three values are indispensable to properly predict the last observed value. Next, to analyze the approach, the absolute prediction errors for the various behaviors in Table 4.4 are computed. Figure 4.20 compares the *MAPS Selective Method*, named in the graph as *selective*, with the minimum absolute prediction error per round (*min*), the average prediction error (*average*), and the prediction errors for parameter combination $\eta = 0.5$ and $\gamma = 0.5$ (as an obvious parameter choice). This series is named 0.5/0.5 in the graph. Since the maximum prediction error is very high it is not plotted in the graph to allow for the better illustration of the other methods.

As shown in Figure 4.20 the Selective Method is in all behaviors almost as good as *min* that means that the approach selects the appropriate parameters. The combination 0.5/0.5 does not really deliver satisfying error rates although this could be an obvious choice of parameters and is often recommended. As expected, *average* presents higher prediction errors even when compared against 0.5/0.5. This is caused by the fact that most parameter combinations result in high error rates. Looking at the different behaviors, the *MAPS Selective Method* accurately predicts the real values for the *LinInc* and *LinDec* behaviors because these data series are the simplest without any trend.

Algorithm 1 MAPS Selective Method.

```
1: input: values $x_1, \dots, x_{n-1}$
2: input: function $DES_{\eta,\gamma}$
3: output: parameter $\eta$, $\gamma$
4: output: prediction value $\hat{x}_n$
5: $error_{min} := \infty$
6: for $i = 0.0$ to $1.0$ do
7:   for $j = 0.0$ to $1.0$ do
8:     $\hat{x}_{n-1,\eta,\gamma} := DES_{i,j}(x_1, \dots, x_{n-2})$
9:     $error := |\hat{x}_{n-1,\eta,\gamma} - x_{n-1}|$
10:    if $error \leq error_{min}$ then
11:      $error_{min} := error$
12:      $\eta := i$
13:      $\gamma := j$
14:    end if
15:  end for
16: end for
17: $\hat{x}_n := DES_{\eta,\gamma}(x_1, \dots, x_{n-1})$
```

In addition, Figure 4.20 shows the average over all eight behaviors. Here, the novel method performs even better than *min* which might seem counter-intuitive at a first glance. Notice however that in the MSM case the average prediction error over all publishers is measured while *min* refers to the best *global* parameter combinations of parameters per publisher, averaged for all publishers. This is caused by the fact that a per publisher parameter combination yields to the best possible prediction. If the errors of the *MAPS Selective Method* are combined, the individual best combination is almost reached and this is still better than the global *min*. Notice also the proportion between predicted values and prediction errors. Since the observed values range from 0 to 600 average prediction errors higher than 50 or 100 are not acceptable. In contrast, the selective method shows satisfying prediction regarding the predicted value range.

In the next section, an alternative approach for parameter setting is presented and subsequently, Section 4.5 evaluates MSM in terms of usability in the MAPS filtering system.

4.4.3 An Alternative Approach

In this section, an alternative approach to improve the prediction of double exponential smoothing is described. Usually, to select the most appropriate values for the parameters η and γ, each peer would investigate the parameter influence using a test collection with a centralized computation, and this training phase would estimate the parameters. Obviously, each peer would have one fixed combination for all publisher peers. In this case, MAPS would not be able to recognize the individual publishing behavior of peers. Therefore, this training approach helps selecting a local parameter combination for each querying peer but is not as flexible as the Selective Method which is able to individually recognize and adopt to the publishing behavior of every peer that is a candidate to monitor.

All alternative ways to select parameter values for η and γ suffer from the the problem of selecting the appropriate training set. Even if distributed approaches are considered, the wrong choice of text collections may result in unsatisfying results.

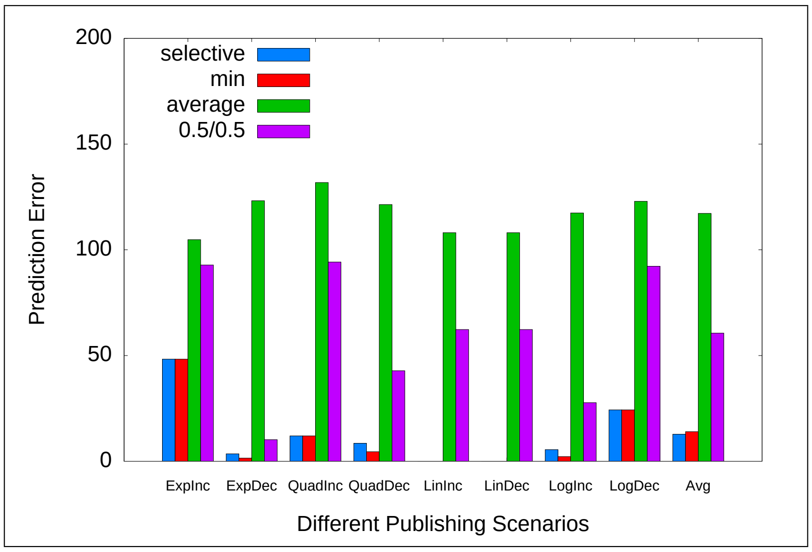


Figure 4.20: Comparing MSM For Different Publishing Scenarios.

4.5 Experimental Evaluation

This section evaluates the Selective Method implemented for the MAPS system. The following sections explain the experimental setup (Section 4.5.1), measures (Section 4.5.2), and data (Section 4.5.3). In Section 4.5.4 the detailed experimental results using different peer publishing scenarios (*Mixed*, *ExpInc*, and *QuadDec*) are presented. Section 4.5.5 summarizes the experimental results regarding MSM.

4.5.1 Experimental Setup

In all experiments described below, the following steps are executed. The network is set up and the underlying *distributed hash table* (DHT) is initiated. Next, there are four *bootstrapping* rounds where the subscriber peers collect the posted directory statistics to the requested continuous queries. After this *bootstrapping phase*, in the six subsequent rounds, the querying peers subscribe to selected publisher peers, to reach a total of ten rounds. During these six rounds that represent the *monitoring phase*, a publisher peer is *monitored* by a subscriber, when the subscriber's continuous query is stored in the publishers local query database. The monitored peers notify the subscriber for all published documents matching the stored continuous query.

All peers in the network publish documents during both phases and at certain intervals (or rounds), the continuous queries are repositioned. The intervals and the number of published documents per round depend on the individual peer behavior that will be inspected in Section 4.5.4. At the end of each round of the subscription phase, the subscriber peers rank publishers using the formula described in Section 4.2 and reposition their queries accordingly. In this experimental setting, *resource selection* is ignored to emphasize the prediction benefits. Thus, in Equation 4.1, $\alpha = 0.0$ is utilized such that only *behavior prediction* is considered.

4.5.2 Experimental Measures

To evaluate the retrieval effectiveness, the measures *recall* is used as the ratio of the total number of notifications received by subscribers to the total number of published documents matching subscriptions. The average recall over all rounds of the subscription phase is reported.

Similar to the publishing behaviors shown previously in the analysis of double exponential smoothing, the recall of the MAPS Selective Method is compared with the minimum and maximum recall that would be possible if a single *global* pair of parameter values is used for η and γ for all the peers in the network. In addition, an *oracle peer selection* approach (referred to as *oracle*) is applied such that it always predicts the accurate IR statistics. The recall of *oracle* represents the highest possible recall that could be achieved by MAPS. The *random peer selection* approach is implemented only for comparison purposes and demonstrates the performance of a random baseline approach.

Besides recall, the *prediction quality* is also analyzed. The average absolute prediction error per key, peer and round is measured. In the graphs MSM is compared only to the minimum prediction error opponent (*min*) since the maximum prediction error (*max*) is very high. Notice that, similarly to the case of recall measures, the *min* and *max* opponents refer to globally selecting the set of parameters that would minimize or maximize the prediction error. The two other approaches are not considered because *random* does not utilize a prediction-based peer selection and *oracle* has by definition a prediction error of zero. In Section 4.5.4 different peer publishing behaviors are investigated in terms of recall and average prediction error.

4.5.3 Experimental Data

The document collection that was used for the experiments contains more than 2 million documents from a focused Web crawl by the focused crawler *BINGO!* [STSW02]. All documents are categorized in one of ten categories: *Music*, *Finance*, *Arts*, *Sports*, *Natural Science*, *Health*, *Movies*, *Travel*, *Politics*, and *Nature*. The category size ranges from 68,000 to more than 325,000 documents (see Table 4.2). There are more than 500,000 different keys (stop words are not considered) and the documents from different categories are used to categorize the peer set. In all experiments, the network consists of 100 peers with 10 peers per category. Using the document collection, seven strong representative single-key queries are extracted: *music*, *arts*, *sports*, *travel*, *hotel*, *offer*, *city*. Single-key queries have the advantage that there is a direct dependency between correctly predicting values and recall. In the case of multi-key queries, there can be the effect that a peer publishes a lot of documents containing the single keys but only a few containing the whole query key set. For simplicity, only single-key queries are considered in this experimental setting. Multi-key queries are considered in Chapter 5.

4.5.4 Experimental Results

After explaining the setup and the dataset of the experimental evaluation, the recall and absolute prediction error results for different peer behaviors are presented. In the set of experiments shown in this section, the publishing behaviors listed in Table 4.4 are utilized. This means that a peer following the *LogInc* behavior publishes in the first round no documents and in the last of the ten rounds 600 documents. In addition, a constant publishing behavior is considered where a peer constantly publishes 300 documents per round during both publishing phases.

To investigate the effectiveness of the MAPS Selective Method, three different scenarios are analyzed. In the first scenario, all mentioned behaviors are used such that some peers have a constant publishing behavior and others increase or decrease their publication rate accordingly. The second scenario looks at the results when all peers in the network have an *ExpInc* behavior, and in the last scenario, all peers follow a *QuadDec* publishing behavior.

The graphs for all scenarios illustrate the performance of the different opponents. The *oracle* opponent shows the maximum recall MAPS can reach by accurately predicting the IR statistics. Naturally, the prediction error for the *oracle* opponent is zero. The *random peer selection* shows the results when publisher peers are selected completely at random. In the random setting, the prediction error is not of interest, because prediction is completely ignored. The *min* and *max* opponents present the best and worst recall MAPS can get with a *global* setting of parameters across *all* peers. Similarly to Section 4.4.2, in the graphs only the *min* absolute prediction error is included to better illustrate the differences between the different opponents. The MSM approach shows the recall and prediction error of the local parameter computation that is used to adapt the DES parameters.

4.5.4.1 The Mixed Publishing Scenario

Over the ten rounds of this scenario, all 100 peers publish together $285,960$ documents following the different publishing behaviors. Figure 4.21 shows the recall depending on the percentage of monitored peers in the network (ρ). As it can be seen, the MAPS Selective Method performs marginally better than the *max* opponent and slightly worse than the *oracle*, whereas *min* and *random* achieve significantly lower recall. This means that the correct choice for the parameters can greatly affect the recall level. Notice that in this scenario the Selective Method not only performs slightly better than the best global parameter choice for η and γ, but also this performance is not greatly affected by the percentage of monitored publishers. This shows that a local auto-adjustment of prediction parameters is possible and results in recall similar to approaches that require global knowledge to compute these parameters. Thus, MSM is able to achieve recall of 60% by monitoring only 20% of all publisher peers in the network.

Additionally, Figure 4.22 shows the absolute prediction error per key, peer, and round. The prediction error of the MAPS Selective Method is about as good as the minimum absolute prediction error computed *min* using a global parameter setting. The error varies for both strategies from 13 to 20 on average. Notice, that *oracle* and *max* are for the known reasons not included in this graph.

4.5.4.2 The ExpInc Publishing Scenario

Figures 4.23 and 4.24 show the recall and absolute prediction error results of the second scenario where all 100 peers follow a *ExpInc* publishing behavior. The total number of published documents amounts to $175,600$. In this scenario, recall is almost independent of the parameter choice (see Figure 4.23). All approaches (*min*, *max*, *oracle*, and *selective*) perform almost the same. Only the *random* strategy is not competitive. Even the prediction quality of *selective* and *min* are similar and vary from about 22 to 35 per key, round, and peer. The absolute prediction error for *max* is still very high and not included in Figure 4.24, but there is no influence to the recall. This is caused by the fact that all publisher peers follow the same behavior *ExpInc*. Compared to other behaviors, this scenario shows the highest absolute prediction errors since the data series shows a strong growing trend.

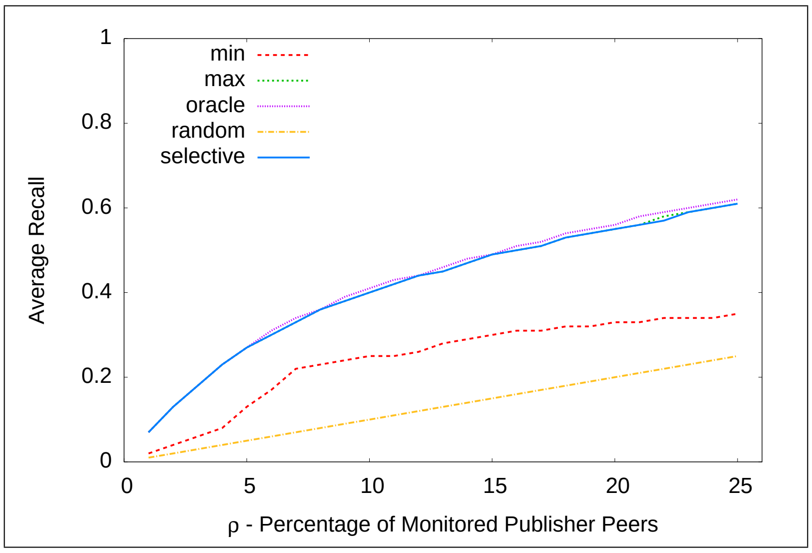


Figure 4.21: Average Recall in *Mixed* Publishing Scenario.

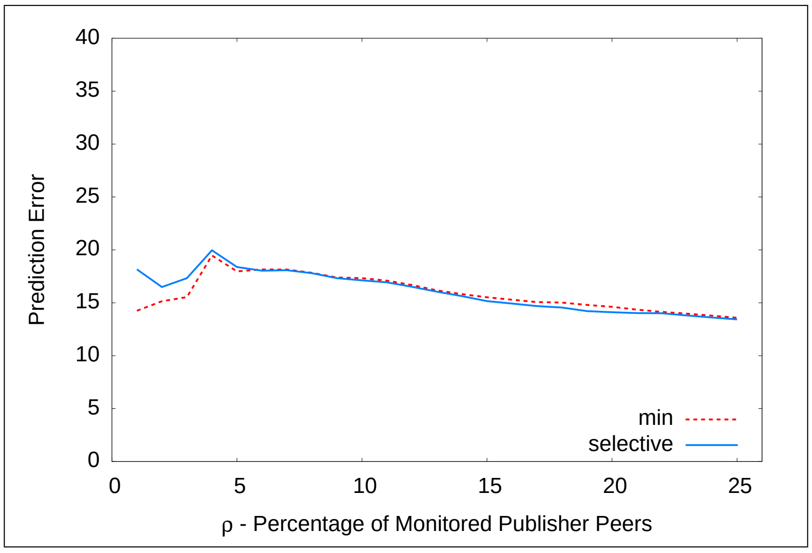


Figure 4.22: Average Prediction Error in *Mixed* Publishing Scenario.

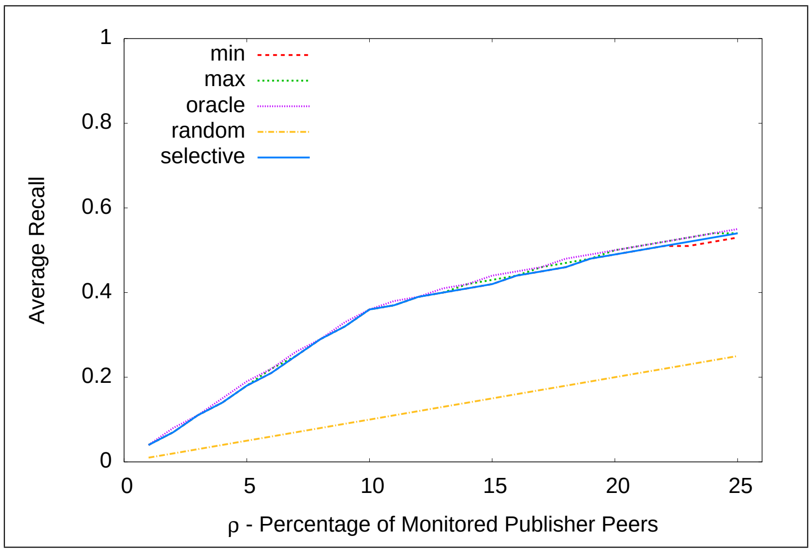


Figure 4.23: Average Recall in *ExpInc* Publishing Scenario.

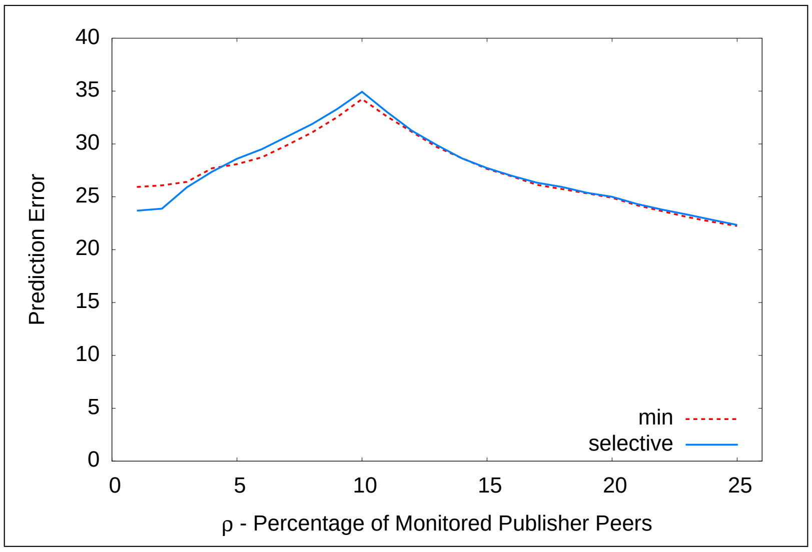


Figure 4.24: Average Prediction Error in *ExpInc* Publishing Scenario.

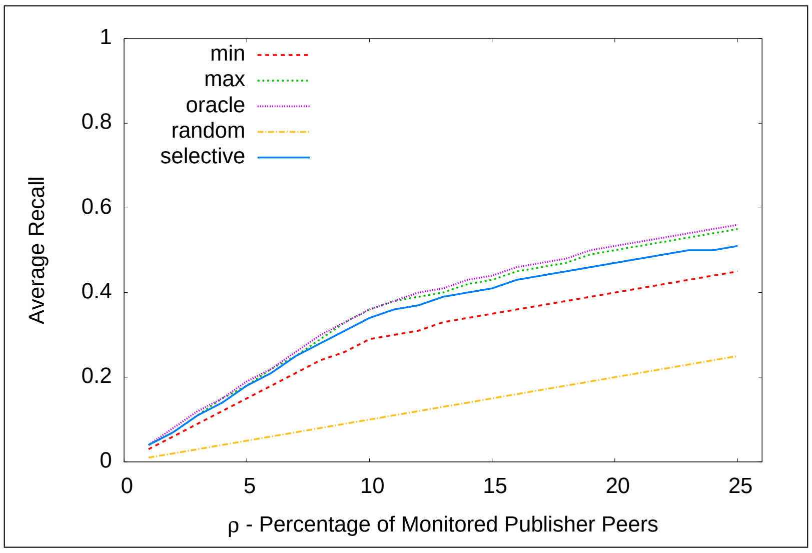


Figure 4.25: Average Recall in the *QuadDec* Publishing Scenario.

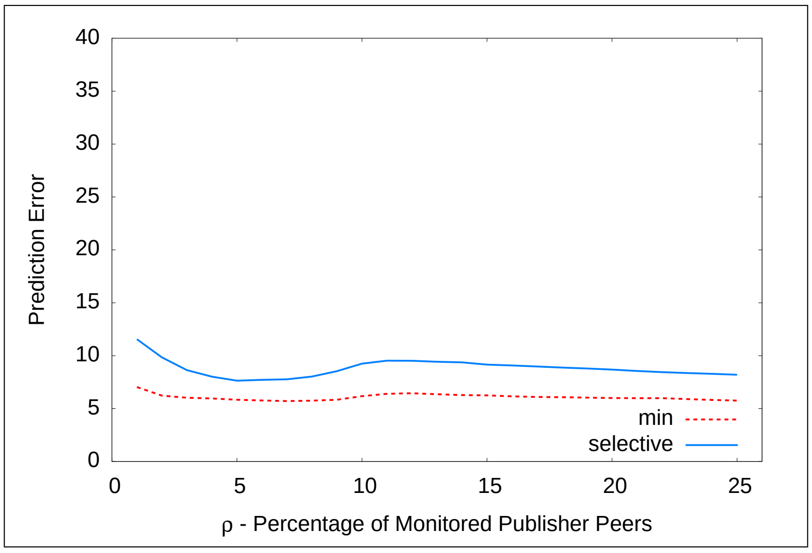


Figure 4.26: Average Prediction Error in the *QuadDec* Publishing Scenario.

4.5.4.3 The QuadDec Publishing Scenario

The last investigated scenario considers the case that all peers publish documents using a *QuadDec* behavior. The overall number of published documents is about $231,000$. Here, the MAPS Selective Method performs much better than *min* in terms of recall but does not reach the level of *max* (see Figure 4.25). This result corresponds to the observations of Section 4.4.2. The same result holds for the absolute prediction errors where *selective* causes higher errors than the best global parameter choice. As also shown in the graph, the *selective* approach still performs better than the worst parameter choice for η and γ. It can also be constituted that the absolute prediction error is almost independent from the number of monitored peers in the network since there is only a small error variation recognizable in Figure 4.26.

4.5.5 Summing Up

The experimental section presented the effectiveness of the MAPS Selective Method in terms of recall improvement against several opponents. In the most considered scenarios, the proposed novel method reached almost the same recall level as the best global parameter combination providing the smallest prediction error. When compared to the opponents examined, the main advantage of MSM is the fact that no additional communication is needed to adopt the two prediction parameters.

4.6 Discussion

This chapter presented a novel peer selection approach, called MAPS, appropriate for supporting approximate publish/subscribe functionality in a structured P2P environment. The peer selection strategy is embedded in the service and protocol architecture presented in the previous chapter. A combination of *resource selection* and *behavior prediction* techniques is applied to the collected metadata such that the most promising publisher peers can be selected. The extensive experimental evaluation approved the effectiveness and efficiency of MAPS in several settings using real Web data.

This chapter also conducted an approach to improve the proposed prediction method of *time series analysis* with *double exponential smoothing*. The MAPS Selective Method clearly improved prediction by individually recognizing publisher behavior without any additional communication overhead. In the next chapter, two strategies to consider correlated keys in MAPS will be presented.

Chapter 5

Correlated Key Sets

Based on the MAPS architecture and peer selection strategies presented earlier, this chapter provides two novel algorithms for exploiting correlations among keywords in continuous queries for an approximate information filtering setting [ZTW08]. This setting assumes that metadata is maintained in a distributed directory, usually on a per-keyword basis, thus disregarding possible relatedness among keywords. The work presented in this chapter extends traditional query routing techniques and strategies from the domain of distributed information retrieval.

Section 5.1 introduces the issue of correlated *keys* (or *keywords*), mentions the main contributions of correlation awareness, and investigates some previous work in this research area. The *baseline approach*, a slightly modified protocol for publish/subscribe, and the correlation measure are presented in Sections 5.2 and 5.3. The two algorithms *USS* and *CSS* to exploit correlations among *key sets* are explained in Section 5.4 and evaluated in Section 5.5. This chapter will revisit the *DV estimation* synopses (*hash sketches* and *KMV synopses*) listed in Section 2.5. The chapter is concluded with a closing discussion in Section 5.6.

5.1 Introduction

In an *approximate* information filtering as the one presented in Chapter 3, only a few carefully selected, specialized, and promising publishers store the user query and are monitored for new publications. Thus, the user query is replicated to these sources and only published documents from these sources are forwarded to the subscriber. The system is responsible for managing the user query, discovering new potential sources and moving queries to better or more promising sources. Since in an IF scenario the data is originally highly distributed residing on millions of sites (e.g., with people contributing to blogs), approximate IF seems an ideal candidate for such a setting. This is also supported by the fact that exact IF functionality has proven expensive for such distributed environments [TIK05a, TX03, AT05]. Thus, approximate IF achieves much better scalability of such systems by trading faster response times and lower message traffic for a moderate loss in recall.

Publisher selection in approximate IF regarding a given continuous query with multiple keywords is driven by statistical summaries (metadata) that are stored by the system. These summaries are provided to the directory by the publishers and can be managed in different ways ranging from centralized solutions like servers or server farms, to super-peer or pure peer-to-peer solutions in the form of a distributed P2P directory built on top of a DHT [Abe01, SMK$^+$01] or other kinds of overlay networks. For scalability, the summaries have publisher granularity, not document granularity, thus capturing the best publisher for certain *keywords* (also referred to as *keys*) but not for specific documents.

This, together with per-key organization of the directory that disregards keyword correlations (also referred to as *key sets*) are two of the basic reasons that may possibly lead to insufficient recall. On the other hand, considering statistics for *all* possible key sets is clearly not possible due to the explosion in the feature space.

As an example scenario, consider a user Bob who wants to follow the discussion about the *presidential elections*[1] in the *US*, and wants to receive notifications from a number of different sites like news agencies, portals, and user blogs. Clearly, Bob would be interested in monitoring a variety of publishers but is not interested in receiving all the articles published by all sources, as it would be the case for exact IF. Thus, in an approximate IF scenario, Bob would submit the continuous query *US presidential elections* to the filtering system. The basic approach would decompose the continuous query into the three individual keys and use the statistics from the directory to compute a combined score (e.g., intersection or some other kind of aggregation of individual key scores) for each key and publisher. This score would represent the probability of each source to publish documents about US presidential elections in the near future. This approach may lead to poor filtering quality as the top-ranked publishers for the *complete* query may not be among the top selected publishers. In the worst case, a selected publisher may deliver many documents for each single keyword, but no single document matching *all keywords*, since this information is not present in the directory.

In this chapter, two approaches that use correlations among keys to improve filtering quality in the scenario described above are introduced and evaluated in detail. The first algorithm (coined *USS* for *Unique Synopses Storage*) uses existing single-key synopses stored in the directory to estimate the publishing behavior of information sources for key sets, while the second (coined *CSS* for *Combined Synopses Storage*) enhances the directory to explicitly maintain statistical metadata about selectively chosen key sets. Both algorithms build upon previous work in the area of *information retrieval* over distributed P2P networks [MBN$^+$06] and extend it to the approximate IF setting in various novel ways.

5.1.1 Contributions

The main contributions of the algorithms presented in this chapter are as follows:

- In contrast to distributed IR settings for one-time searching where sources are ranked according to their document collections (i.e., using *resource selection* strategies), in approximate IF the publishers are ranked according to their probability to publish relevant documents in the near future, which poses different requirements for the maintenance of statistics. This is the first work to develop algorithms for exploiting keyword correlations in such an dynamic IF setting.
- The self-limited approach of [MBN$^+$06] for two-key queries is extended to the case of multi-key continuous queries for an arbitrary number of keys. Next, new algorithms to approximate multi-key statistics by combining the statistics of arbitrary subsets are provided.
- *Hash sketches*, used in [MBN$^+$06] for compactly representing the documents, yield inaccurate results when considering continuous queries with more than two keys. Thus, the usage of recent state-of-the-art techniques for compact representation of multisets is proposed [BHR$^+$07] and applied. These new techniques (called *KMV synopses*) allow the system to compute accurate synopses for multi-key queries, and further improve the filtering effectiveness.

[1] The United States presidential election of 2008 is scheduled for November 4, 2008

Symbol	Explanation
$\|X\|$	number of distinct documents in a multiset X
D	set of documents in the system
D_i	set of documents on publisher p_i
a, b	individual keys
ab	key set (both of a and b)
$D(a)$	set of documents in D containing key a
$D_i(a)$	set of documents in D_i containing key a
$df(a)$	frequency of key a in D $(= \|D(a)\|)$
$df_i(a)$	frequency of key a in D_i $(= \|D_i(a)\|)$
$SYN(a)$	synopsis representing documents in $D(a)$
$SYN_i(a)$	synopsis representing documents in $D_i(a)$
$d(a)$	directory peer responsible for key a

Table 5.1: Summary of Notation

5.1.2 Previous Research on Correlated Key Sets

This section present some previous work in the context of information retrieval and filtering that includes the usage of correlations among keys within these settings. All methods mentioned in this thesis organize the statistics about publisher peers, which drive the necessary query routing decisions, on a per-key basis disregarding key correlations. The only recent works that consider key correlations in the context of P2P search are [MBN+06] and [SLZ+07]. [MBN+06] considers frequent key combinations in query logs and documents for P2P IR, where [PRL+07] proposes a framework for *Highly Discriminative Keys* (HDK), which includes correlated key combinations; however, it does not give any algorithms for managing the corresponding statistics in a distributed setting and for correlation-aware query routing. Previous work on information filtering does not consider correlations among keys so far such that the work in this thesis presents the first approach.

5.2 The Baseline Approach

This section presents the *baseline approach* as a starting point to introduce the two algorithms that exploit correlations in key sets. The baseline approach uses the protocols presented in Section 3.3 with an approximate IF architecture consisting of three different system components: the *directory*, the *publishers* and the *subscribers*. The notation that will be used is summarized in Table 5.1.

In this chapter, the additions and modifications over the baseline protocols presented in Chapter 3 are described including the distribution of special-purpose synopses by the publishers. The synopses represent the inverted lists of documents a publisher hosts concerning a key a. This can be done using *hash sketches* or *KMV synopses* as presented in Section 2.5. A publisher p_i inserts an identifier for each document contained in its collection into a local hash sketch or KMV synopsis for key a to obtain $SYN_i(a)$. The distributed directory stores all disseminated statistics. Since there is a well-defined directory peer responsible for each key (through the DHT hash function), the synopses representing the index lists of all publishers for a particular key a are *all* sent to the same directory peer $d(a)$. Thus, $d(a)$ can compute a moving-window estimate for the global df value of a – $df(a)$ – by performing an union operation for all synopses $SYN_i(a)$ sent by every publisher p_i for key a.

The baseline approach intersects the statistics for the single keys of a continuous query $cq = \{k_1, k_2, \ldots, k_n\}$ consisting of multiple keys k_i, and sends the continuous query only to (a subset of) the publishers that published (or will publish) statistics for *all* queried keys. However, this approach may lead to reduced recall, since a publisher appearing in the publisher lists for both keys k_i and k_j will not necessarily publish documents containing both k_i and k_j. Thus, to select an appropriate publisher for cq, the statistics of the *key set* have to be considered to determine more accurately its future publishing behavior. Obviously, the larger the subset of cq statistics are maintained for, the more accurate the prediction about the behavior of the publisher will be.

5.3 Correlation Measures

This section covers the question how to measure the correlation among keys. Thus, the existing measures for correlated key pairs is presented and extended for capturing *relatedness* among keys in key sets. Next, the *correlation model* that will drive the extended synopses construction and publisher selection is explained.

In [MBN+06], the *conditional probability* that a random document contains a key a given that it contains a key b was introduced as an asymmetric measure of relatedness. Using this measures, there is no need for knowing or estimating the total number of documents, since the estimator for key sets with two keys a and b is given by Equation 5.1 for the conditional probability $\hat{P}(A|B)$ as follows:

$$\hat{P}(A|B) = \frac{df(ab)/|D|}{df(b)/|D|} = \frac{df(ab)}{df(b)} \tag{5.1}$$

To estimate the value of $df(ab)$, the number of documents containing keys a and b, *hash sketches* and *KMV synopses* provide appropriate multiset operations. A nice property of this measure is that it can be applied without knowing (or estimating) the number of documents $|D|$. Subsequently, [MBN+06] proposes methods to identify sufficiently frequent and interesting key pairs, and select those that present the lowest correlation. The output of this process would then be used to guide query routing to appropriate peers for local result retrieval.

Notice that standard measures like the *correlation coefficient* has the drawback that its estimation requires knowledge (or an estimate) of the total number of documents in the network. Moreover, there may be situations where it is important to capture that key b is related to key a, but the reverse direction is uninteresting. For example, in popular, recent Web queries the key *Olympics* often implies that the same query contains also the key *Beijing*[2], but the reverse direction has a much weaker association from a user viewpoint. So, in the proposed setting, the conditional probability is a better measure for relatedness.

Here, the approach summarized above is modified to consider arbitrary numbers of keys (instead of only key pairs) and to identify appropriate key sets in an approximate information filtering scenario. Thus, to consider an arbitrary number of keys in a correlated key set $S = \{k_0, k_1, \ldots, k_{n-1}\}$ and to compute the probability estimator that a random document contains k_0 given that it contains all other keys, the previous formula has to be modified as follows:

$$\hat{P}(K_0|K_1 \ldots K_{n-1}) = \frac{df(k_0 k_1 \ldots k_{n-1})}{df(k_1 \ldots k_{n-1})} \tag{5.2}$$

[2]since the Olympic Games 2008 take place in Beijing

Notice that, in an information filtering setting, the continuous queries are actually long-standing queries that will be assessed several times in the future. Thus, all continuous queries can be effectively considered as candidate key sets for harvesting multi-key statistics. In contrast, a requested query in the retrieval scenario can be asked only once. Should a more selective system behavior be intended (e.g., for performance reasons), the submitted continuous queries can be further analyzed using *frequent itemset mining* techniques [AIS93, FSGM$^+$98] to discover the most common ones.

To identify either uncorrelated key pairs or negatively correlated key pairs, additional statistics are needed to find the best publishers to index a multi-key continuous query. To clarify this need, consider a key pair ab that has no correlation. For these keys, there are only few publishers in the network that have the potential to publish in the future relevant documents that would contain both a and b, and these publishers cannot be located by selecting and combining the statistics for key a and key b alone. The conclusion from this is that a pair ab has to be considered as interesting if both $\hat{P}(A|B)$ and $\hat{P}(B|A)$ are *below* some threshold β within the documents. Extending the above to a key set S with an arbitrary number of keys using the conditional probability in Equation 5.2, for each key k_0 the value of $\hat{P}(K_0|K_1 \dots K_{n-1})$ has to be estimated. The key set S is of interest if all estimated probabilities are *below* some threshold β.

5.4 Exploiting Correlations

This section presents two new algorithms to exploit correlations among key sets: (i) the USS algorithm (Unique Synopses Storage) presented in 5.4.1 uses single-key synopses (hash sketches or KMV synopsis) already stored in the distributed directory to estimate the filtering behaviors for key sets, and (ii) the CSS algorithm (Combined Synopses Storage) explained in 5.4.2 enhances the single key directory to explicitly maintain statistical meta-data about selectively chosen sets of multiple keys depending on the correlation measures (conditional probability) introduced in previous Section 5.3. Both algorithm build-upon and extend the *baseline algorithm* introduced in Section 5.2.

5.4.1 The USS Algorithm

In this section, the USS algorithm (Unique Synopses Storage) is presented. USS uses single-key synopses already stored in the distributed directory to estimate the publishing behavior for complete key sets. As the distributed statistics of a publisher p_i regarding a key a contain the synopsis $SYN_i(a)$ (to represent the documents of p_i containing key a), it is possible to compute the number of documents containing all keys of a given key set S. For this, the algorithm uses the intersection operation for multisets as explained in detail in Section 2.5 as an estimation for the frequency of the whole key set in p_i's local collection ($df_i(S)$). Together with prediction techniques presented in this thesis, the most promising publishers can be selected by applying appropriate scoring functions based on publisher behavior prediction.

To explain the algorithm, consider a subscriber s subscribing with a multi-key continuous query $cq = \{k_1, k_2, \dots, k_n\}$ containing n distinct keys. According to the USS algorithm the following steps are executed:

1. For each key $k_j, 1 \leq k \leq n$ in cq, subscriber s contacts the directory peer $d(k_j)$ responsible for k_j and retrieves the statistical summaries published individually for all keys, including the synopses $SYN_i(k_j)$ produced by a publisher p_i.

2. For each publisher p_i appearing in all statistics, s computes an estimation for $df_i(cq)$ using formulas introduced in Section 2.5 and applies prediction techniques based on *time series analysis* to compute a behavior prediction score as described in Chapter 3. This score indicates p_i's potential to provide appropriate published documents in the future.

3. Subscriber s sends the continuous query cq to the top-ranked publishers with the highest publisher scores. Only these publishers will index cq and in the future notify s about matching documents. Obviously, non-selected publishers will not notify s for published matching documents, since they are not aware of cq.

4. Due to publisher churn and dynamics in publishing behavior, s has to reposition the continuous query cq by repeating steps 1 to 3 in a periodic way.

The USS approach has the major advantage that it can be performed for *all possible* key sets and queries in the directory, while the CSS approach, presented in the next section, uses multi-key statistics of judiciously selected key sets, and can only be applied to these *predefined* key set collections. Below, some important issues about the USS approach are listed:

- **Higher Network Load**: To exploit the single-key statistics for a given key set, the directory has to send long lists of statistics to the requesting subscriber since the lists have to be merged. In contrast, given multi-key statistics in the directory, only the statistics of the top-ranked publishers have to be retrieved. Beyond this, the number of directory peers in the USS algorithm depends on the number of keys contained in the continuous query.

- **Inaccuracy**: While both hash sketches and KMV synopses allow to estimate the number of distinct values in the intersection of multisets, it is inaccurate to create a synopsis for the intersection that represents the documents containing all keys. In the case of hash sketches, an estimation for a key set with more than two keys suffers from a significant degrading in accuracy due to its indirect way to compute the *sieve formula*, whereas the KMV synopses provide better cardinality estimations for multiple set intersections.

- **Prediction Errors**: Considering single-key statistics to predict publisher behavior introduces additional errors. The subscriber has to perform time series analysis of estimated values thus increasing the probability of prediction errors whereas prediction on accurate values shows very promising results.

5.4.2 The CSS Algorithm

To overcome the problems of the USS algorithm, the CSS algorithm (Combined Synopses Storage) is introduced. The algorithm identifies valuable key combinations, enhances the directory to explicitly maintain these multi-key statistics, and exploits these statistics to improve publisher selection. In addition, the CSS algorithm combines multi-key statistics for subsets of the requested continuous query.

As already discussed in Section 5.3, uncorrelated or negatively correlated keys are of interest to collect multi-key statistics. The CSS algorithm aims at addressing the following questions: (i) how to decide which key sets are of interest to disseminate additional multi-key statistics; (ii) how to disseminate multi-key statistics of publishers to the directory; (iii) how to exploit the additional statistics to improve publisher selection; (iv) how to deal with multi-key statistics for subsets of the complete continuous query.

5.4.2.1 Assessing Key Sets

Given a key set $S = \{k_1, k_2, \ldots, k_n\}$ with n distinct keys, a deterministic function is employed (e.g., by selecting the lexicographically highest or lowest key) to select one directory peer that has to assess the relatedness among the keys of S and be responsible for this key set. A deterministic approach with pre-hashing the key-values ensures load balancing among the directory peers. The CSS approach uses the following three steps to assess a candidate key set S where $d(S)$ is the directory peer that is responsible for the assessment decision:

1. Initially, $d(S)$ contacts all directory peers responsible for the other keys $k_j \in S$ to retrieve the synopsis $SYN(k_j)$ representing all documents in the network containing the key. The answering directory peers $d(k_j)$ for a key k_j compute locally the synopsis $SYN(k_j)$ by using the union operation over all individual publisher synopses $SYN_i(k_j)$ (see Section 2.5).

2. Subsequently $d(S)$ computes the intersections among the synopses $SYN(k_j)$, to retrieve the estimated cardinality of documents containing all keys in the candidate set (denoted as $df(S)$).

3. Finally, using Equation 5.2, directory peer $d(S)$ then computes the conditional probabilities for each key $k_j \in S$.

Small values for the conditional probabilities show that the occurrence of all keys in the documents are largely independent, meaning that publisher selection decisions can strongly benefit from the existence of available multi-key statistics. Thus, CSS initiates the creation of these additional summaries if the conditional probabilities for all keys $k_j \in S$ are below some threshold β.

To further optimize message traffic between directory peers, a threshold α is introduced, such that if the conditional probability of a key k_j is above α, publisher selection strategy does not have to consider the single-key statistics of that key. The idea behind this is that k_j is contained in *almost all* documents containing the rest of the keys $S \backslash \{k_j\}$, making this key set a candidate for multi-key statistics. The experimental evaluation will also investigate this strategy in Section 5.5 by computing the conditional probabilities of several continuous query.

5.4.2.2 Disseminating Multi-Key Statistics

As soon as a key set has been assessed as a useful candidate that is worth collecting multi-key statistics for (as explained before), publishers have to learn this fact immediately. Especially in an information filtering scenario, where statistics have to get updated, the continuous process of directory refreshment can be used to disseminate the knowledge about valuable multi-key sets.

Assuming that a directory peer $d(S)$ that is responsible for a key $k \in S$ and key set S identified key set S as useful. According to the algorithm, $d(S)$ is also responsible to maintain the multi-key statistics for the complete key set S. Whenever a publisher p updates its statistics for key k, $d(S)$ can inform the publisher about the new key set. In this way, p will compute its local statistics for the complete set S and disseminate them to the directory peer $d(S)$. This procedure does not incur any additional messages compared to the *baseline approach* described in Chapter 3, since additional statistics can be piggybacked on messages that need to be sent anyway.

5.4.2.3 Exploiting Multi-Key Statistics

Next, it is assumed that a subscriber s submitting a continuous query cq asks the directory peers responsible for the keys in cq to provide the related statistics. Since cq is included in the request message, a directory peer $d(S)$ that stores statistics about a key set S, where $S \subseteq cq$, will return the statistics for S together with the single-key statistics it is responsible for. The following section explains how to compute the statistics for a multi-key query by combining statistics from subsets available in the directory. Subsequently, the subscriber collects the multi-key statistics and uses them to perform publisher ranking. Note that the subscriber applies publisher selection based on multi-key statistics, and is thus able to predict the publishing behavior of sources more accurately. Contrary, the baseline algorithm performs a separate prediction for each single key. If no summaries for the key set (or subsets, see Section 5.4.2.4) have been published, the baseline procedure using single-key summaries is still applicable without contacting additional directory peers.

5.4.2.4 Combining Multi-Key Statistics

A usual case for continuous queries with more than two keys is that there are no statistics for the full key set S of the query, because, e.g., the assessment step did not consider the full key set as sufficiently useful. However there might be available statistics for several multi-key *subsets* of S. In this case, by design, all of these subsets can be found by the subscriber by simply asking the directory peers responsible for the single keys as explained in the previous section.

To give an example, assume that the directory maintains the multi-key statistics of $S_1 = \{a, b, c\}$ and $S_2 = \{b, c, d\}$. Then, for a given five-key query $S = \{a, b, c, d, e\}$, the algorithm has several options at hand. The CSS approach proposes to select all *maximal* subsets among the available multi-key summaries. This is efficient in terms of network costs because the entire continuous query will be sent to all single-key directory peers anyway. But this consideration opens up a space of optimization strategies; Combining such incomparable but mutually related multi-key statistics is reminiscent of the recent work on multidimensional histograms with incomplete information [MMK+05], but the setting here has the additional complexity of very-high-dimensional key space.

To combine the different multi-key statistics, the CSS algorithm weights them depending on the size of the multi-key subset. The scoring function to calculate a publisher score is given by Equation 5.3:

$$score_S(p) = \sum_{S_i \subseteq S} |S_i| \cdot predScore_{s_i}(p) \tag{5.3}$$

Here, S_i is a subset of the key set S contained in the continuous query, $|S_i|$ is its cardinality, and $predScore_{s_i}(p)$ represents the likelihood that p will produce a document containing S_i in the future. This score is produced using time series analysis techniques similarly (plus resource selection techniques) as presented in Chapter 4. Thus, to obtain a publisher score, the system sums-up all prediction scores for the subsets S_i that have been received from the directory peers such there is no S_j with $S_i \subset S_j$.

The intuition behind weighting the prediction score with the size of the key set $|S_i|$ is that the prediction score for small subsets dominates the sum. This happens because the number of documents containing all the keys of small key sets is higher, resulting in higher prediction scores. Next section experimentally investigates different scenarios where multi-key statistics for the full continuous query are not available, but a combination of the statistics of subsets is possible.

5.5 Experimental Evaluation

In this experimental section, the USS and CSS algorithms are evaluated using two real-life corpora with Web and blog data. Since it is extremely difficult to find real-life continuous queries except by obtaining proprietary data (e.g., from CNN's news or Springer's journal alert system), the experiments utilize the popular Web queries contained in the *Zeitgeist* query-log, and treats them as subscriptions (e.g., assuming that a user is interested in staying informed about his favorite pop star, or a big sport event). The *Google Zeitgeist*[3] query-log is published periodically for different geographic regions to express the most interesting search trends.

5.5.1 Experimental Setup

The evaluation compares the filtering performance of both algorithms with different synopses (*hash sketches* and *KMV synopses*) and against a *baseline* algorithm that computes publisher scores using single-key frequencies to predict the publishing behavior of information producers. This baseline algorithm corresponds with the *baseline approach* presented in 5.2. The prediction mechanisms have an average prediction error of 10% to ensure a realistic system behavior.

As quality measure to evaluate the filtering performance of both algorithms, the experiments use *recall*. Recall in the IF setting is defined as the ratio of the total number of notifications received by subscribers to the total number of published documents matching the subscriptions. In the experiments, the *average recall* over several publication rounds is considered. In all graphs illustrated in this section, recall depends on the percentage of monitored publisher peers.

5.5.2 Experimental Data

For the experimental setup, a recently proposed *benchmark* [NBMW06] designed specifically for use in the evaluation of distributed and P2P settings is utilized. Briefly, the benchmark consists of more than $800,000$ Web documents drawn from the *Wikipedia* corpus, and an algorithm to distribute the documents among $1,000$ publishers with controlled overlap that mimics the real-world behavior.

The *Zeitgeist* query-log contains queries with one, two, and three keys. To investigate filtering quality depending on the conditional probabilities, the set of distinct single-keys in all queries is used to create two-, three, and four-key queries by combinations. During query evaluation, a continuous query is indexed in up to 25% of the publishers. The experiments use the concept of publication rounds to simulate the time properties of the published documents; here, a publisher peer publishes around 400 documents in each round, which could represent the weekly publications of a digital library or news portal.

The second data set focuses on blog data[4]. Out of a huge set of blogs, about $1,000$ blogs have been selected that published more than 300 postings each during a time period of three weeks. The selected blogs have been crawled and assigned to $1,000$ publishers. The crawled blog data also spans over three weeks. Again, the same *Zeitgeist* query-log is used to evaluate the filtering performance, while the query repositioning algorithm was executed to simulate one query repositioning per week.

[3]http://www.google.com/press/zeitgeist.html archives query-logs since 2001
[4]Another well-known source of blog data is Blogsphere [BK07].

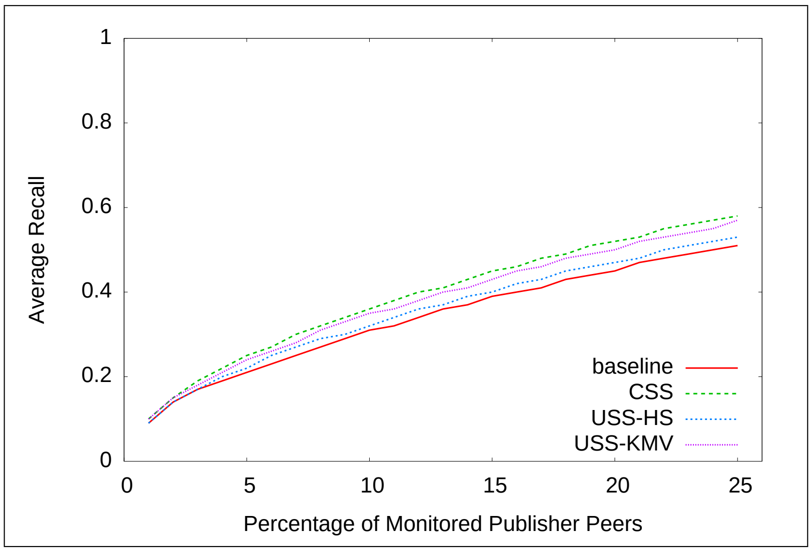


Figure 5.1: Average Recall for Two-Key *Zeitgeist* Queries.

5.5.3 Experimental Results

This section shows the experimental results of the evaluation for both data sets. Section 5.5.3.1 considers experiments using the Web data collection whereas Section 5.5.3.2 uses the blog data collection.

5.5.3.1 Results Using Web Data

Figure 5.1 (respectively 5.2) shows the filtering quality for all two-key (respectively three-key) continuous queries from the Web query-log using the Web data set. The evaluation compares a *baseline* publisher selection algorithm based on behavior prediction for single-key statistics as described in this thesis in Chapter 3, with the CSS approach of multi-key statistics maintained in the directory, and with the USS approach using two different synopses (*hash sketches* and *KMV synopses*).

The results show the recall improvements obtained by the proposed algorithms. For two-key queries, 24% publishers have to be monitored to reach a recall level of 0.50 in the baseline approach, whereas, using the CSS approach, the subscriber only has to monitor 19% of the network. The CSS outperforms both USS approaches, because it offers more accurate and explicit statistics for the key-pairs. Comparing the two USS approaches, the use of KMV synopses slightly improves filtering quality when compared to hash sketches.

Considering the results for the three-key query set, the improvements are much higher. The CSS approach reaches a recall of 0.79 by subscribing to 15% of all publishers. In contrast, the baseline approach reaches a recall of 0.44 by monitoring the same number of publishers. Using this query set, a considerable difference between the results of the two USS approaches can be seen. The USS-KMV approach that uses KMV synopses almost reaches the result quality of the multi-key approach, while USS-HS suffers from the inaccuracy of combining hash sketches of more than two sets. Recall that KMV synopses use a direct way to intersect multisets, whereas hash sketches reside on an indirect computation that causes loss in accuracy.

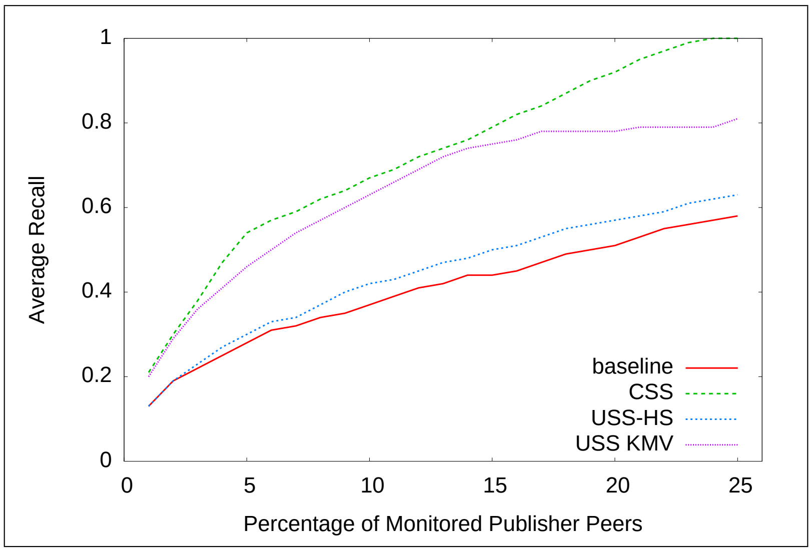


Figure 5.2: Average Recall for Three-Key *Zeitgeist* Queries.

To investigate the filtering quality of both approaches for four-key continuous queries, queries were created using the single keys from the real-world *Zeitgeist* query set. Out of the full set of all possible four-key queries, the 500 combinations with the highest number of matching documents in the Web collection have been selected. Examples of resulting queries are *time war unit world*, *day nation world game*, or *war world london british*. All selected continuous queries have more than $3,000$ matching documents within the Web data collection.

In the first experimental series with this query set, the general filtering quality of the CSS and USS approaches are investigated. Figure 5.3 shows that CSS outperforms the baseline approach and both USS approaches. When hash sketches are used to represent documents the filtering performance is worse even from the baseline approach due to the inaccuracy of multiset operations. Hash sketches have to use the *sieve formula* to perform the multiset operation thus suffering from inaccurate estimations. Contrary, the use of KMV synopses improves the filtering quality because it guaranties better distinct-value estimations for the intersections of multiple sets (see Section 2.5.2). Notice that the selected four-key queries do not fully show the improvement margins of the CSS algorithm, since more selective key sets with less matching documents benefit more from multi-key statistics.

Using the set of four-key continuous queries described above, the evaluation measured the filtering quality for combining multi-key statistics as proposed in Section 5.4.2.4. Figure 5.4 illustrates the improvements for several scenarios of available multi-key statistics in comparison to the baseline approach. The filtering quality of CSS gives the upper bound by using the exact multi-key statistics of the full continuous query:

- In the first scenario *CSS-1*, all four possible three-key statistics are available to use select the most appropriate publishers. This means for a four-key continuous query *abcd* that the statistics for *abc*, *abd*, *acd*, and *bcd* are available. Notice that multi-key statistics for single keys or pairs can also be stored in the directory but not be used since the proposed algorithm in Section 5.4.2.4 to combine multi-key statistics only considers *maximal* subsets.

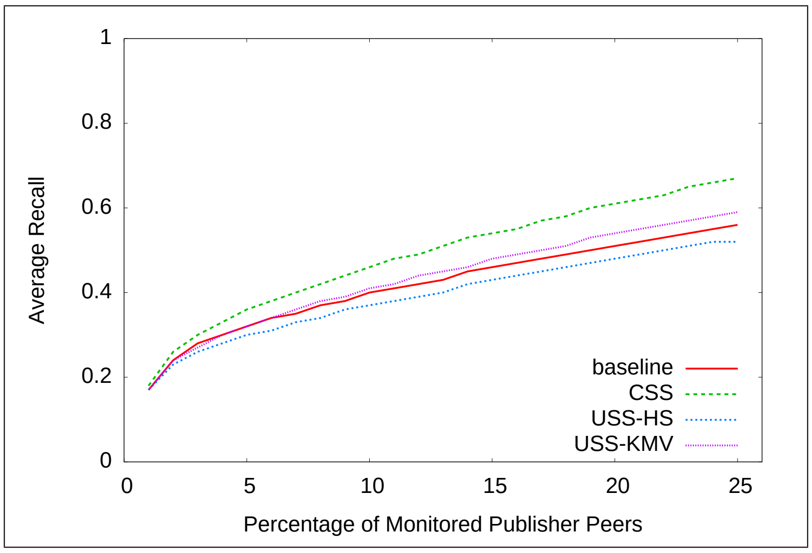


Figure 5.3: Average Recall for Four-Key *Zeitgeist* Queries.

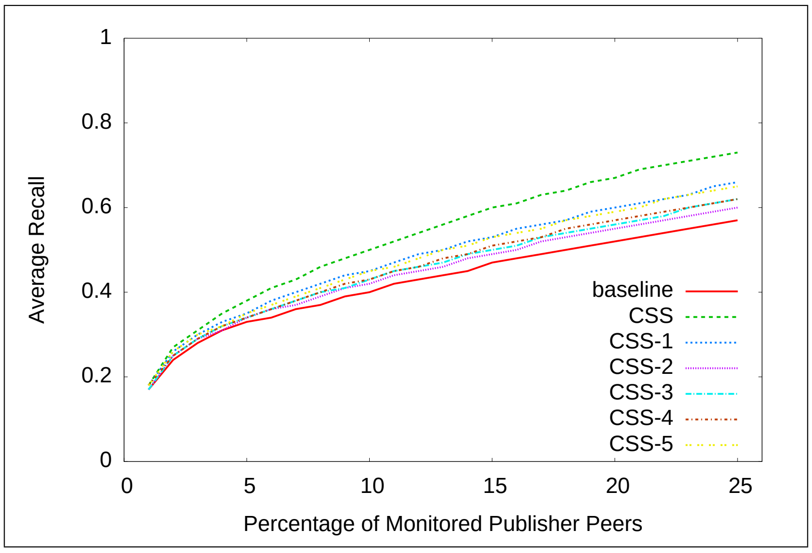


Figure 5.4: Average Recall for Different Multi-Key Statistics.

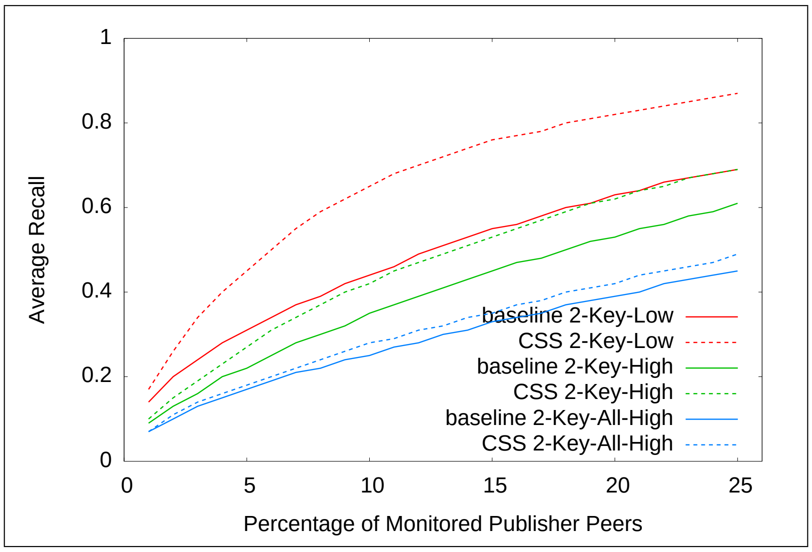


Figure 5.5: Average Recall Improvements for Two-Key Queries.

- All six two-key statistics are provided by the directory in the second scenario denoted as *CSS-2* but the directory does not store statistics for three key subsets.
- The third *CSS-3* scenario assumes the existence of two two-key statistics such that the whole key set is covered (e.g., for query *abcd*, the statistics of *ab* and *cd* are available).
- One three-key plus one complementary single-key statistics are provided by the directory in the *CSS-4* scenario (e.g., statistics for *acd* and *b*).
- The last *CSS-5* scenario assumes two overlapping multi-key statistics for one two-key plus one three-key statistics. Here, the multi-key statistics for *abc* and *cd* meet the scenario condition.

The results show that all combinations improve the baseline approach, but cannot reach the filtering quality of the CSS approach. Naturally, filtering quality depends on the size of the multi-key statistics: the scenarios (*CSS-5* and *CSS-1*) where multi-key statistics for three keys are considered to select publishers show the best filtering performance; in contrast, the three other scenarios only have two-key statistics.

Figures 5.5 and 5.6 demonstrate another series of experiments that targeted the improvements in filtering performance over the baseline algorithm, when using the proposed correlation measures for multi-key statistics. To conduct this experiment, all possible two- and three-key queries are created using the single-keys from the *Zeitgeist* query-log. Three different query sets with a size of 100 each are selected; the query sets have the following properties:

- *Low-Corr* includes those key sets where all keys have a low conditional probability such that there are only a few documents in the Web data set that contain both keys at the same time.

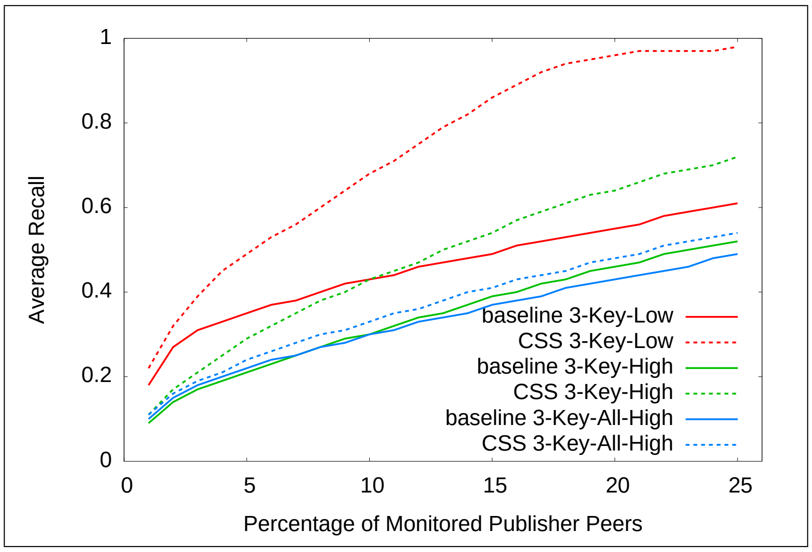


Figure 5.6: Average Recall Improvements for Three-Key Queries.

- *High-Corr* consists of key sets with high conditional probability for *at least one* key. The main characteristic of such a key set is that one of the keys almost only appears in documents that also contain the remaining keys.
- *All-High-Corr* consists of key sets with high conditional probabilities for all keys. Here, the key set appears together in documents very often.

Tables 5.2 and 5.3 show example two-key and three-key queries for the used query sets described above. Underlined scores denote high conditional probabilities. The threshold depend on the number of keys. The key sets *island brother* and *british sport star* show low conditional probabilities whereas in the key sets *football sport* and *nation unit world*, all keys have a high conditional probability. The other two examples belong to the query set *High-Corr* where keys show a diverse correlation measure (*at least* one key has a high conditional probability).

key_A, key_B	$P(A\|B)$	$P(B\|A)$
island brother	0.10	0.05
time gift	$\underline{0.62}$	0.03
football sport	$\underline{0.33}$	$\underline{0.27}$

Table 5.2: Relatedness of Example Two-Key Queries.

key_A, key_B, key_C	$P(A\|BC)$	$P(B\|AC)$	$P(C\|AB)$
british sport star	0.13	0.21	0.22
time face friend	$\underline{0.83}$	0.25	0.22
nation unit world	$\underline{0.54}$	$\underline{0.47}$	$\underline{0.45}$

Table 5.3: Relatedness of Example Three-Key Queries.

As already explained in Section 5.4, subscribing to key sets where all keys have low conditional probabilities yields to the highest filtering result improvements when monitoring the same percentage of publishers. This is caused by the fact that a low conditional probability means that there are a lot of matching documents this key without matching the remaining keys. Thus, when monitoring only 10% of the publishers, *2-Key-Low-Corr* has an recall improvement from 0.44 to 0.65 whereas *2-Key-All-High-Corr* only improves filtering quality from 0.25 to 0.28 (see Figure 5.5). Similar results hold for three-key queries in Figure 5.6 where *3-Key-Low-Corr* has an improvement from 0.43 to 0.68 compared to improvement for *3-Key-All-High-Corr* from 0.30 to 0.33. This leads to the conclusion that the CSS algorithm has no significant effect for key sets where all keys are highly correlated, while it significantly improves filtering for key sets with low correlations.

Analyzing the results for key sets where one key is highly correlated to all others (*High-Corr*), an smaller improvement compared to unrelated keys can be observed. Here, the use of the CSS approach is possible, but an alternative strategy is proposed: a high conditional probability for a key k means that there is almost no additional information included in multi-key statistics for the full key set in comparison to the key set without k. As an example, the multi-key statistics for key set *time face friend* yield to similar filtering results than the multi-key statistics for *face friend* because the conditional probability $P(time|face, friend)$ is high (83% of all documents containing *face* and *friend* also contain *time*). The same observation holds for the key set *time gift* where *time* denotes the highly correlated key. There, 60% of all documents containing *gift* also contain *time*.

5.5.3.2 Results Using Blog Data

In this section, experimental results from the blog data experiment are presented. Overall, the observations for blog data lead to similar conclusions with the Web data corpus, with CSS outperforming USS and baseline.

The results for all two-, and three-key queries from the *Zeitgeist* query-log are presented in Figure 5.7. The complete *Zeitgeist* queries are listed in the appendix B. Again, the KMV synopses perform better than hash sketches. The USS-KMV approach is almost as good as the the upper bound of CSS whereas USS-HS at least outperforms the baseline approach. Notice that, contrary to the Web data collection, a smaller number of blogs have to be monitored to acquire a sufficiently high recall. This happens because all blogs are highly specialized and thus only a small number of them contribute new posts matching the requested continuous queries. Compared to this, the Web data collection provides less specified publisher peers.

Subsequently, Figures 5.8, 5.9, and 5.10 illustrate the average recall improvements for two-key, three-key queries, or four-key queries respectively considering the different query sets introduced in the previous section (*Low-Corr*, *High-Corr*, and *All-High-Corr*). Again, the experiments use selections of all possible key set combinations created by single keys with maximal numbers of matching blog posts. The sizes of all query sets range from 10 to 20, and now, key sets with four keys are regarded, too.

Similar to the experiments with Web data, the CSS approach for multi-key queries with low conditional probabilities shows the highest recall improvement. In Figure 5.8, there is almost no improvement for two-key queries where both keys have a high conditional probability (*2-All-High-Corr*) but the improvement for queries with low conditional probability (*Low-Corr*) is significant: from 0.53 to 0.67 when monitoring 10 blogs. Thus, blog data experiments confirm the previous observations. In Figures 5.9 and 5.10, the differences between low correlated (*Low-Corr*) and high correlated key sets (*All-High-Corr*) are reduced but still existing.

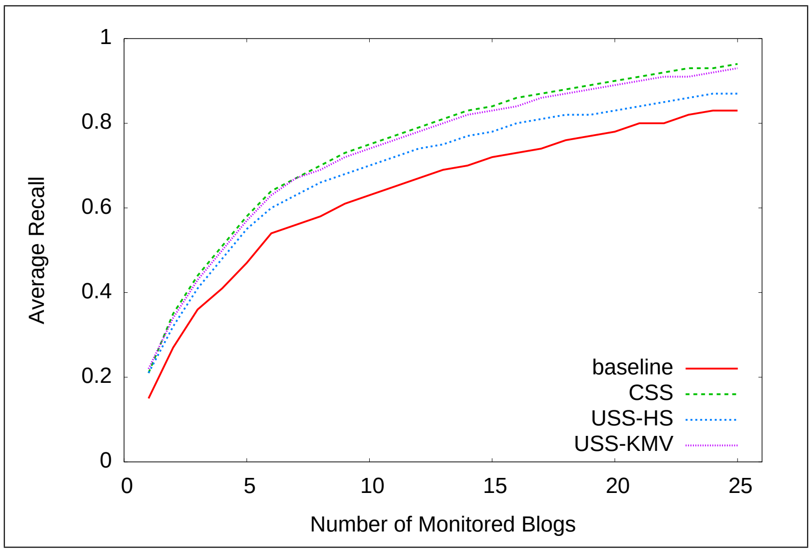


Figure 5.7: Average Recall for Two- and Three-Key Queries.

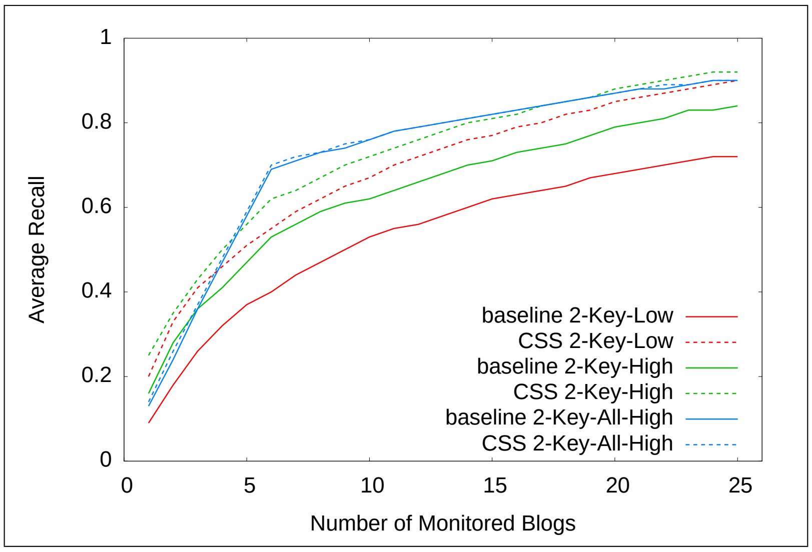


Figure 5.8: Average Recall Improvements for Two-Key Queries.

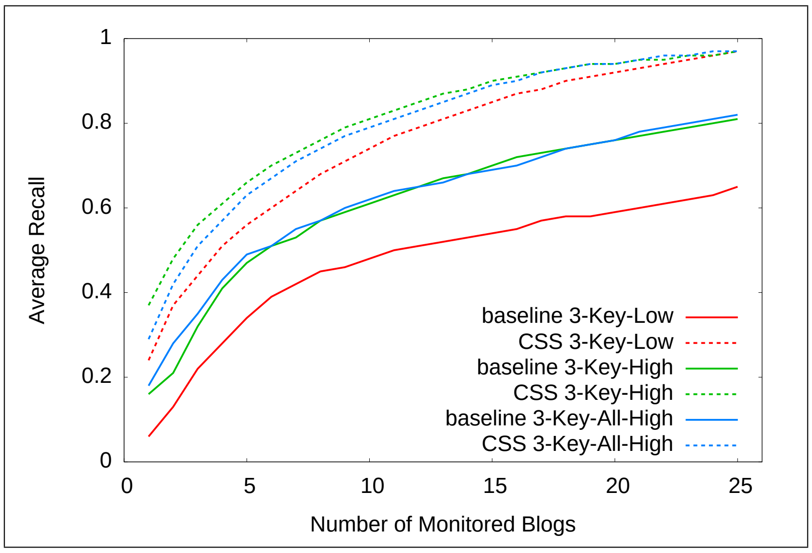


Figure 5.9: Average Recall Improvements for Three-Key Queries.

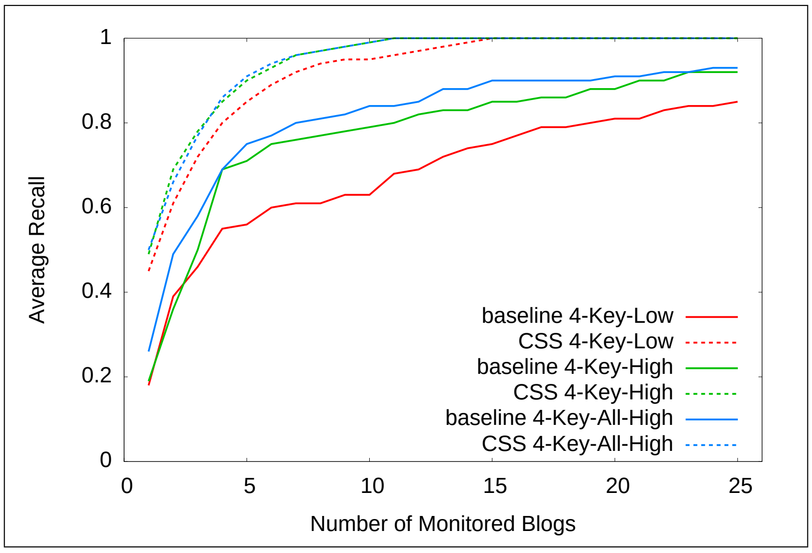


Figure 5.10: Average Recall Improvements for Four-Key Queries.

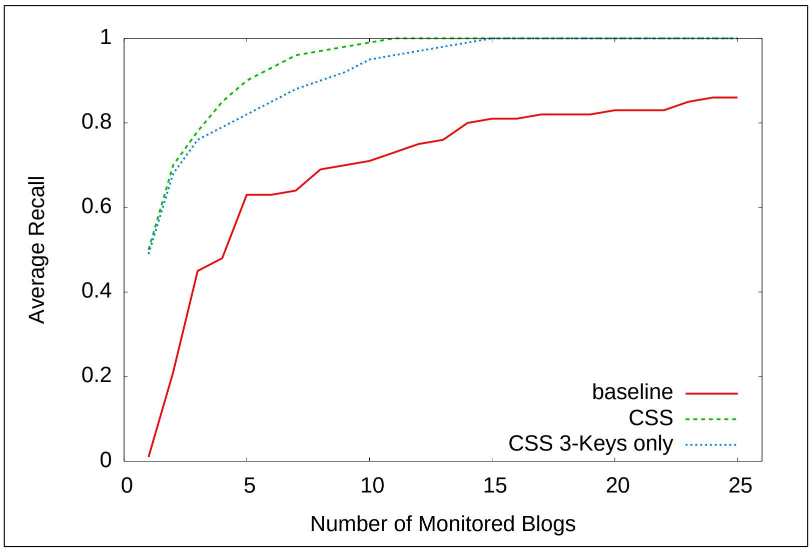


Figure 5.11: Average Recall for High-Correlated Four-Key Queries.

Similar to the experiments with Web data, the next evaluation investigates under which conditions, the statistics of subsets are sufficient to get a satisfying recall. Thus, Figure 5.11 uses four-key queries where one key is highly correlated with the remaining set. Formally, there is $k_0 \in S = \{k_0, k_1, k_2, k_3\}$ such that $\hat{P}(k_0|k_1, k_2, k_3)$ is very high. The graph shows the recall for the query using CSS approach for the complete key set k_0, k_1, k_2, k_3 in comparison to the results applying CSS approach with key set k_1, k_2, k_3 missing k_0. Obviously, the average recall for 10 four-key queries with this characteristics is almost stable. The graph also shows the average recall for the baseline approach. This leads to the following conclusion: the CSS approach can be applied to subsets missing a high correlated key because there is no additional information when considering the multi-key statistics of the complete key set.

key_A, key_B, key_C, key_D	$P(A\|BCD)$	$P(B\|ACD)$	$P(C\|ABD)$	$P(D\|BCD)$
london time train bbc	<u>0.97</u>	0.55	0.36	0.34
day face news london	0.51	0.27	0.26	0.17

Table 5.4: Examples of Relatedness among Four-Key Queries.

The following example makes this point clear using the conditional probabilities in Table 5.4. Using a multi-key continuous query *london time train bbc*, it is sufficient to apply the CSS approach for the subset *time train bbc* because the key *london* has a conditional probability of 0.97. In other words, 97% of all posts containing the key *london* also contain the remaining keys. So, there is no significant additional information included in the multi-key statistics of the complete key set.

5.6 Discussion

This chapter introduced two approaches that utilize correlations among keys to improve filtering quality. The USS algorithm uses existing single-key synopses stored in the directory to estimate the publishing behavior of information sources for key sets, while the CSS algorithm enhances the directory to explicitly maintain statistical metadata about selectively chosen key sets. This is the first work to develop algorithms for exploiting keyword correlations in such an dynamic IF setting.

Whereas the approach of [MBN+06] is limited to two-key queries, the USS algorithm can be applied to multi-key continuous queries for an arbitrary number of keys. Therefore the usage of very recent state-of-the-art techniques for compact representation of multisets (KMV synopses) is included in the algorithm. These synopses overcome the restrictions of hash sketches. The CSS algorithm uses a new strategy to approximate multi-key statistics by combining the statistics of arbitrary subsets. This strategy is useful when the multi-key statistics for the full continuous query are not included in the directory.

The experimental evaluation illustrated the filtering performance improvements of both algorithms in comparison to the baseline approach presented in this thesis. All experimental series used two different real-world collections for Web and blog data, and applied real-world *Google Zeitgeist* queries. The evaluation also investigated filtering performance gains depending on the introduced correlation measure (*conditional probability*) representing a way to compute the relatedness among keys. The next chapter presents the current prototype implementation of the MAPS approximate information filtering approach.

Chapter 6

Prototype Implementation

This chapter presents the current prototype implementation [ZHTW08] of the MAPS approximate information filtering approach introduced in the previous chapters of this thesis. Therefore, the Minerva search engine developed as explained in [BMPC07] is used and extended with additional components to realize both functionalities (one-time searching and publish/subscribe) in one unifying system.

Section 6.1 introduces some aspects of current Web search. In Section 6.2, the main aspects of the Minerva search engine are explained including an implementation overview. In addition, the most important principles that drive the P2P search paradigm are illustrated. Section 6.3 presents the additional components for approximate pub/sub functionality that have been implemented. This section also shows the usage of both functionalities by following some example (continuous and one-time) queries. Screenshots from the system's graphical user interface (GUI) are used to demonstrate and explain the MAPS system. A short overview of some other prototype systems for IR and IF in P2P networks presented in Section 6.4, and the discussion in Section 6.5 conclude this chapter.

6.1 Introduction

Today, full-fledged Web search is more or less under the control of centralized search engines. In the US, more than 10 billion searches have been conducted at core search engines in November 2007. Here, the big players share the market as shown in Table 6.1[1]. In Germany, *Google* is even more dominant with a market share of almost 90%[2]. Various projects have been started to build and operate a P2P Web search network (e.g., [TD04, LKP+05, CAPMN03, YVGM04]) including the Minerva project. Web search and Internet-scale file content search seem to be perfect candidates for a P2P approach for several reasons:

- The Web is increasing at a much faster rate than the indexing capability of any centralized search engine [HT99, WMYL01, SE00]. In addition, the data is highly distributed *by nature*, residing on millions of sites.
- A P2P network could potentially outperform even the largest server farm in terms of processing power and could, thus, enable much more advanced methods (e.g., ontology-based background knowledge). A P2P Web search engine can potentially benefit from the intellectual input of a large user community, as every peer's behavior is influenced by a human user.
- There is growing concern about the world's dependency on a few *monopolistic* search engines and their susceptibility to commercial interests.

[1] http://www.comscore.com
[2] http://www.webhits.de

	Search Queries (in billion requests)	Share of Searches (in percent)
Total Core Search	10.03	100.0%
Google Sites	5.88	58.6%
Yahoo! Sites	2.25	22.8%
Microsoft Sites	0.98	9.8%
Ask Network	0.46	4.6%
Time Warner Network	0.45	4.5%

Table 6.1: US Search Engine Rankings November 2007.

6.2 The Minerva Search System

This section introduces the Minerva[3] prototype for P2P search [BMWZ05, BMT+05a, BMT+05b]. The next sections elaborate the main principles of the Minerva search engine (Section 6.2.1), present its architecture (Section 6.2.2), and summarize some core fundamentals of its prototype implementation (Section 6.2.3).

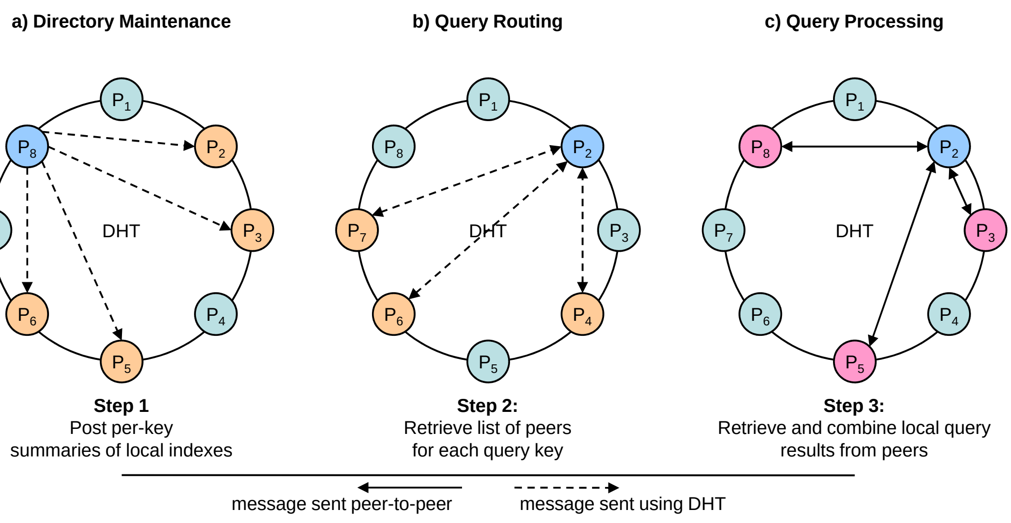
6.2.1 System Principles

In [LLH+03], approaches to comprehensive Web search based on a P2P network have been considered infeasible, or at least being a grand challenge, from a scalability viewpoint. Early approaches typically spread inverted index lists across the directory such that each peer is responsible for maintaining a subset of index lists. Such systems allow for exact and complete execution of top-k style aggregation queries over the P2P network. However, bandwidth requirements and also latency issues raise concerns about their scalability claims. Novel approaches [CW04, MTW05] try to overcome these challenges by utilizing efficient large-scale top-k query aggregation algorithms for distributed systems.

The system design of Minerva differs from the approaches mentioned above. Instead of disseminating inverted index lists across the directory, Minerva uses only pointers to promising peers (enriched with compact statistical metadata describing the index contents of that peer) and utilize these pointers to answer multi-keyword queries. Here, some fundamental design principles are listed that influenced the architecture of the Minerva search system presented in the next section:

- The first principle is peer *autonomy* such that peers do not disseminate index lists to other peers and the network, and peers do not store/maintain index lists of other peers.
- Minerva introduces the principle of keyword-*granularity*. Peers only disseminate statistical metadata concerning single keywords. This avoids the dissemination of full documents to the network.
- The principle of *approximation* means that the Minerva search system does not provide exact and complete answers because only a fraction of peers is contacted to retrieve data.

[3]Minerva is the Roman goddess of wisdom, and also the icon of the Max-Planck Society (and a Greek underwear manufacturer).



a) Directory Maintenance
b) Query Routing
c) Query Processing
P1
P2
P3
P4
P5
P6
P7
P8
DHT
Step 1
Post per-key summaries of local indexes
Step 2:
Retrieve list of peers for each query key
Step 3:
Retrieve and combine local query results from peers
message sent peer-to-peer
message sent using DHT

Figure 6.1: Minerva Search Architecture and Query Execution.

- Minerva does not forward requests to *all* possible peers in the network. The *scalability* principle ensures that only the most appropriate peers are involved in the query execution.

6.2.2 System Architecture

The P2P Web search prototype system Minerva [BMT+05b] assumes a P2P collaboration in which every peer is autonomous and has a local index that can be built from the peer's own crawls or imported from external sources representing the user's interest profile. The local index contains inverted index lists with URLs for Web pages that contain specific keywords. A conceptually global but physically distributed directory, which is layered on top of a *distributed hash table*, holds compact, aggregated information about the peers' local indexes. The DHT partitions the key space, such that each directory peer is responsible for maintaining the metadata about a randomized subset of keys. For failure resilience and availability, the metadata may be replicated across multiple directory peers. The three steps of query execution work as follows (see Figure 6.1):

1. *Directory Maintenance*: Every peer publishes in step 1 a number of key-specific statistical summaries (*posts*) describing its local index to the directory (shown in Figure 6.1a). Posts contain contact information about the peer who published this summary together with statistical information to support appropriate query routing strategies (e.g., the size of the inverted list for the key, the maximum average score among the key's inverted list entries, or various other statistical synopses [MBN+06]). The DHT is used to determine the directory peer responsible for this key.

2. *Query Routing*: If the results of a local query execution are unsatisfactory, the user can utilize in step 2 the distributed directory to identify more promising peers for a particular query as follows (shown in Figure 6.1b): for each query key, the query initiator identifies the peer that is currently maintaining the appropriate statistics, using the DHT functionality. Then the query initiator retrieves the relevant posts

by issuing requests directly to these peers. The statistical synopses contained in the posts are used to perform query routing, i.e., to identify the most promising peers for that particular query.

3. *Query Processing*: After a small subset of peers has been selected, the query is forwarded and executed based on their local indexes in step 3 (shown in Figure 6.1c). Note that this communication is carried out in a *point-to-point* manner. Finally, the remote peers return their local results to the query initiator, where they are combined into a single result list (*result merging*).

This Minerva baseline approach can be extended such that multiple directories are utilized to maintain information beyond local index summaries, such as information about local bookmarks [BMWZ04], information about *relevance assessments* (e.g., from peer-specific query logs or *click streams* [KLFW06]), and (implicit or explicit) *user feedback*.

The upcoming sections discuss the two major challenges of query execution with Minerva, namely *query routing* in Section 6.2.2.1 and *result merging* in Section 6.2.2.2.

6.2.2.1 Query Routing

Query routing is one of the key issues to make P2P search feasible. A user requesting a multi-keyword query expects a high-quality top-10 or top-100 ranked result list. Therefore, the system has to select the peers that have to answer the query. This decision is based on statistical information stored in the directory. [BMWZ05] introduces the baseline query routing approach utilizing *resource selection* strategies (e.g., CORI [CLC95], DTF [NF03], or GlOSS [GGM95]). These resource selection strategies (or *database selection*) originally been designed for distributed IR need to be adapted to the large-scale and the high dynamics of a P2P system. The work presented in [BMT$^+$05a, MBTW06] introduces an overlap-aware query routing approach that takes the novelty of peers into account. Another extension [MBN$^+$06] considers the correlations among keywords to improve P2P query routing. Caching strategies as shown in [ZBW08] can improve result-quality of P2P search and reduce response times by retrieving cached results of previously executed queries. In addition, aggressively reuse cached results of even subsets of a query towards an approximate caching technique can drastically reduce the bandwidth overheads.

6.2.2.2 Result Merging

A key issue in a P2P search engine is *result merging*: the querying peer has to combine all top-ranked local results into a single, comprehensively ranked result list, which is eventually displayed to the user. The obvious solution is to use local statistics but this leads to scores incomparable across peer boundaries. There are several options to solve this problem:

- If all peers agree on a common scoring function that exclusively utilizes *objective* components (e.g., term frequencies), scores are immediately comparable across peer boundaries, and result merging simply involves sorting the combined result list of all peers by document scores.
- A second option combines local query results in a *round-robin* manner but does not really consider the differences between peers and their ability to contribute satisfying results.
- A last option tries to estimate global statistics (e.g., global document frequencies [BMTW06, PMBW05]) and local peers use these estimates to produce compatible document scores.

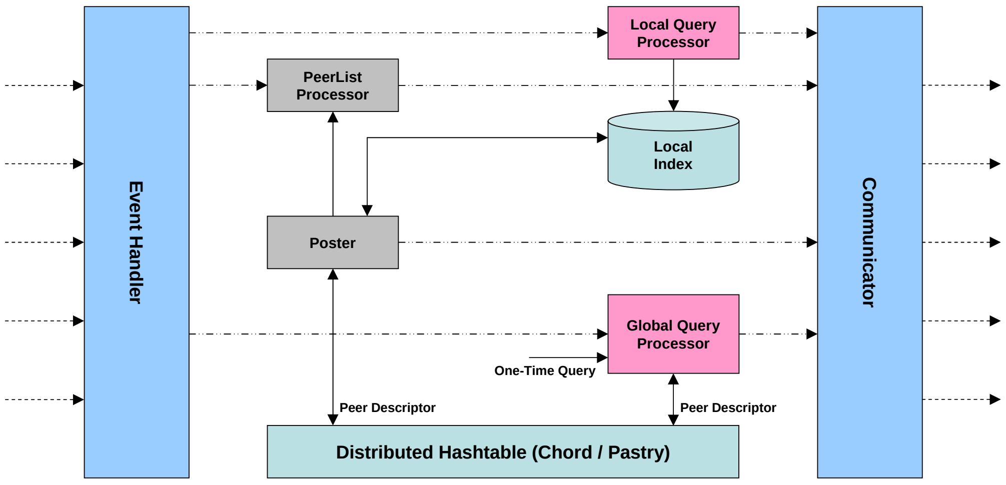


Figure 6.2: Minerva Search Engine Implementation.

6.2.3 Implementation

Minerva is implemented in a platform-independent way using Java 5. The software architecture of a peer is illustrated in Figure 6.2. Each peer is build on top of a globally distributed directory which is organized as a distributed hash table, e.g, *Chord* [SMK$^+$01] or *Pastry* [RD01a]. Minerva utilizes the lookup functionality of the DHT to provide a mapping from keys to peers. Early versions of Minerva relied on a reimplementation of the Chord protocol, and more recent versions run as a *FreePastry* application, using Pastry's network routing mechanisms and Past's storage functionalities to become resilient to network dynamics and peer failures. The latter version works as follows: each peer maintains a *PastryNode*, implementing the *PastryApplication* interface, and is registered at a *PastryEndpoint*. Once registered, the PastryNode delivers incoming messages to the registered applications. There exist two different implementations of *Past* (PastImpl and GCPastImpl). GCPastImpl is an extension of PastImpl that offers garbage collection based on time-stamps. Minerva uses this extended version in order to prune outdated metadata objects after a specific time interval.

The DHT layer returns a *Peer Descriptor* object containing the contact information (e.g., IP address and port) of the peer currently responsible for a key. A *Communicator* is instantiated with this data to perform the communication with remote peers. Each peer runs an *Event Handler* listener that receives incoming messages and forwards them to the appropriate local components of a peer. Every peer has a *Local Index* holding the peer data using any database system capable of executing standard SQL commands (e.g., *Oracle*, *MySQL*, or *Cloudscape/Derby*). The index can be used for query execution by the *Local Query Processor* component. Additionally, the *Poster* component uses the local index to produce the key-specific summaries that are published to the global directory using the *Communicator*. Each peer implements a *PeerList Processor* to maintain the incoming posts, i.e., all posts from across the network regarding the subset of keys that the actual peer is currently responsible for. Notice that *Past* is designed to handle such inserts natively, such that recent versions of Minerva do not need to use a *PeerList Processor*.

When the user initiates a one-time query using Minerva, the *Global Query Processor* component uses the DHT to locate the peer responsible for each query key and retrieves the respective metadata using Communicator components. After appropriately processing the metadata, the *Global Query Processor* forwards the complete query to selected peers, which in turn process the query using their *Local Query Processors* and return their results. In a final step, the *Global Query Processor* merges these remote results and presents the merged result to the user. There are some cases where Minerva peers communicate directly, i.e., without using the DHT lookup functionality.

6.3 The MAPS Filtering Extension

The MAPS approximate information filtering approach introduced in this thesis has been integrated into the Minerva search prototype such that Minerva provides in addition to one-time searching an approximate publish/subscribe functionality in addition. Section 6.3.1 presents the implementation aspects concerning the extension of Minerva, while Section 6.3.2 explains the usage of the extended Minerva prototype by executing example one-time and continuous queries. There, the various parts of the graphical user interface (GUI) are illustrated.

6.3.1 Implementation

In this section, the changes implemented to add the publish/subscribe functionality at the Minerva prototype are explained. Figure 6.3 shows how the three new components of a peer are integrated in the existing system.

Besides one-time queries, the modifications described below, allow issuing continuous queries. Thus, the *Global Query Processor* component is able to handle both types of queries as input. A continuous query has in addition a *lifetime* parameter to determine how long such a request should be valid. To process a continuous query requested by a user, the *Global Query Processor* utilizes the *Time Analysis Storage* component that maintains statistical metadata of active continuous queries. Therefore, the future behavior of publisher peers can be predicted by applying *time series analysis* techniques to stored metadata as described in Chapter 3 in detail.

Collecting publisher metadata is performed similarly to the one-time querying by utilizing the lookup functionality of the DHT, and asking the peers responsible for the keys in the continuous query. Having selected the most promising publisher peers for a query, the *Global Query Processor* uses the *Communicator* to send the continuous query to the remote peers. Whenever a peer receives a continuous query (as an event of the *Event Handler*), the query is stored at the *Continuous Query Store*. This store denotes the second new component for filtering. Each time, a peer inserts a new query, outdated queries stored at the peer are removed from the *Continuous Query Store*.

The third new component to integrate approximate publish/subscribe functionality to Minerva, is the *Publisher* module. This component receives new documents as input and adds these documents to the *Local Index*. In addition, the *Publisher* checks the *Continuous Query Storage* for active continuous queries matching the publication. Subscriptions that are no longer valid (e.g., because the *lifetime* has expired) are removed from the store. Checking the matching continuous queries delivers a set of peers that have subscribed with one of these queries, and have to be notified about the published document. The *Publisher* component sends a notification message using the *Communicator* to inform the subscriber peers about the new document.

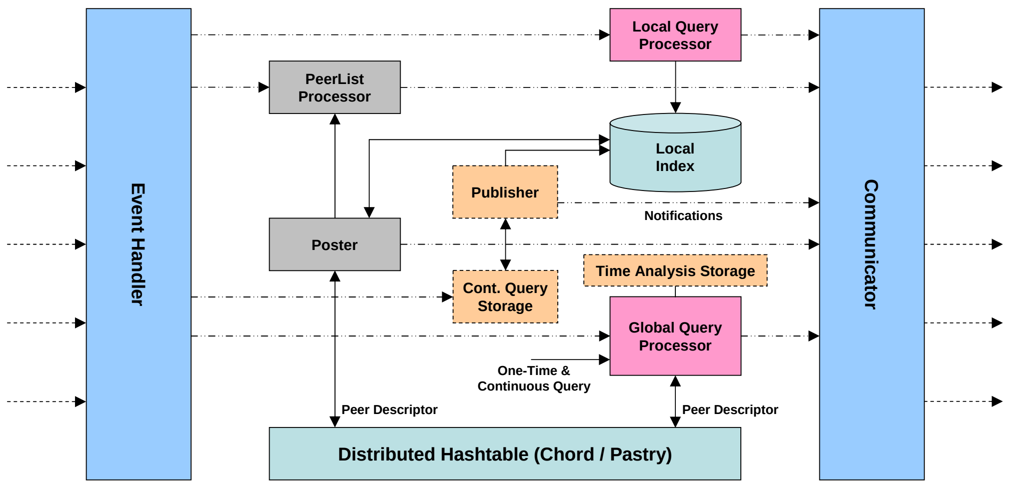


Figure 6.3: Extended MAPS Implementation with IF Components.

6.3.2 Example Use of MAPS

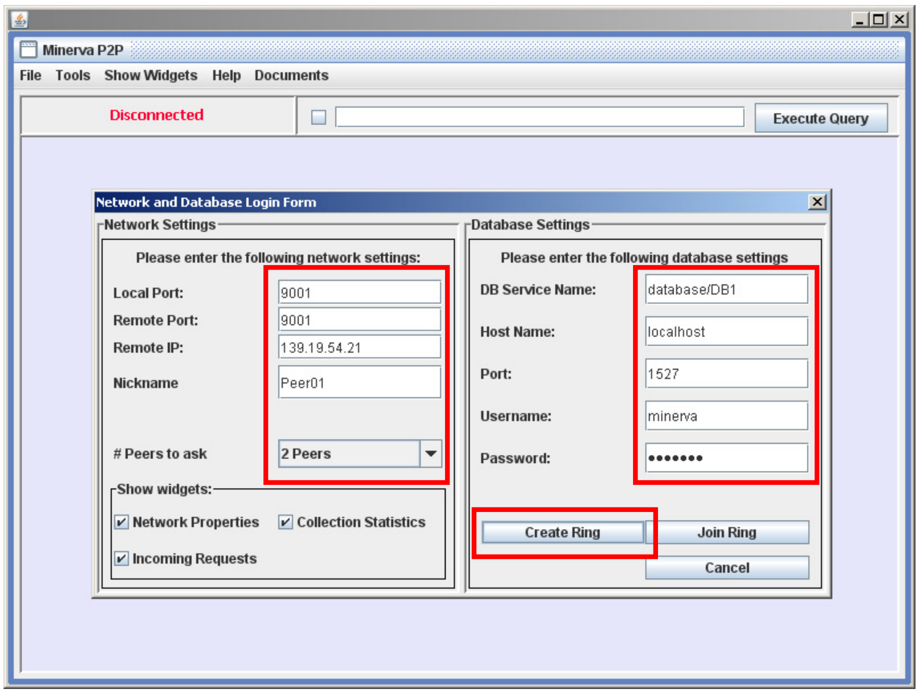
This section showcases the usage of Minerva (respectively MAPS) with the extended approximate publish/subscribe functionality by means of screenshots taken from the latest prototype version. It also serves as a short explanation how one-time and continuous query functionality is used within Minerva.

In this showcase example, 10 *Cloudscape* or *Derby* databases are used to host about 100 documents per database. Thus, 10 peers can be created to manage one of the 10 collections each. To publish new documents, there is an additional database that hosts about 100,000 additional documents simulating the input from a crawling component such as BINGO! [STSW02] commonly used in such a P2P search engine. A peer can select a random or query-specific document from this shared collection to store it in its local database. The shared collection is also realized by a *Cloudscape* database. The peer instances can run on one or more machines. The following sections present a usage scenario and explain the graphical user interface of the client in detail.

6.3.2.1 Minerva Initialization

When starting the Minerva client, the user has to input the network details and database login as shown in Figure 6.4. On the left, the *Local Port* and the *Nickname* of the peer have to be specified. If the peer joins an existing network, the *Remote IP* address and the *Remote Port* number of a random peer in the network must be declared. On the right, the connection to a local database has to be stated including *DB Service Name*, *Host Name*, *Port* number, *Username*, and *Password*. Here, the database *DB1* runs on the same machine realized by the server mentioned in the previous section.

The *Create Ring* button specifies that a new P2P network should be created whereas the *Join Ring* button is used to contact an existing network. The form also allows to select which widgets should be shown automatically after initialization. All three check boxes are preselected.


Minerva P2P
File Tools Show Widgets Help Documents
Disconnected
Execute Query
Network and Database Login Form
Create Ring
Join Ring
Cancel

Network Settings

Please enter the following network settings:

Local Port:	9001
Remote Port:	9001
Remote IP:	139.19.54.21
Nickname	Peer01
# Peers to ask	2 Peers

Show widgets:

☑ Network Properties ☑ Collection Statistics

☑ Incoming Requests

Database Settings

Please enter the following database settings

DB Service Name:	database/DB1
Host Name:	localhost
Port:	1527
Username:	minerva
Password:	•••••••

Figure 6.4: Initiating the Minerva Client GUI.

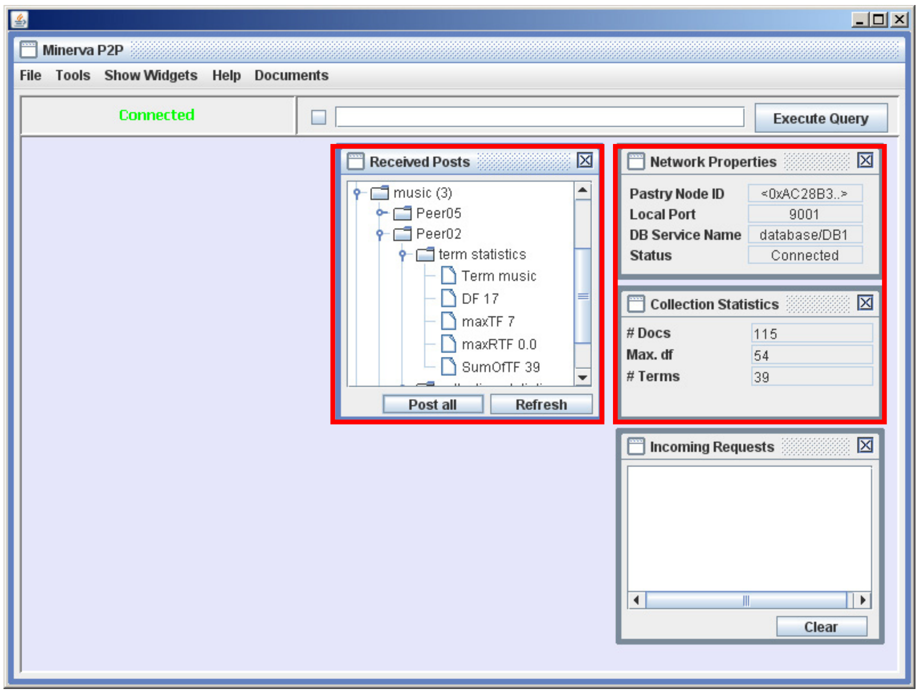
6.3.2.2 Publishing Metadata

After initialization, the peer is connected to a P2P network. Figure 6.5 shows on the right the *Network Properties* and *Collection Statistics*. The *Network Properties* illustrate that the peer is connected, has a certain *Pastry Node ID* with local port, and owns the local collection with *DB1* as *DB Service Name*. The *Collection Statistics* present some statistical information (e.g., the number of documents) concerning the local collection.

The *Received Posts* widget in the middle manages the metadata the peer is responsible for. The list shows all keys a peer stores metadata for, e.g., three peers in the network have published metadata for key *music*. The entry of *Peer02* tells that this peer hosts 17 documents containing the key *music*. The *Refresh* button updates the shown list such that metadata received in the meantime from other network peers is updated. The *Post all* button starts the posting procedure of this peer and distributes the metadata of its local collection to the network. In addition, the *Collection Statistics* are recomputed to incorporate new documents that have been recently published.

6.3.2.3 One-Time Query Execution

Figure 6.6 summarizes the one-time query execution. The query input field contains the requested query *modern music*, and the unselected check box designates that this is a one-time query. When the user executes the query, the peer contacts the directory peers storing the metadata for key *modern* and key *music* to retrieve the key statistics. For this query, the peer itself hosts the key statistics such that it is fact that only *Peer02* stores documents for both keys.



music (3)
Peer05
Peer02
term statistics
Term music
DF 17
maxTF 7
maxRTF 0.0
SumOfTF 39

Pastry Node ID	<0xAC28B3..>
Local Port	9001
DB Service Name	database/DB1
Status	Connected

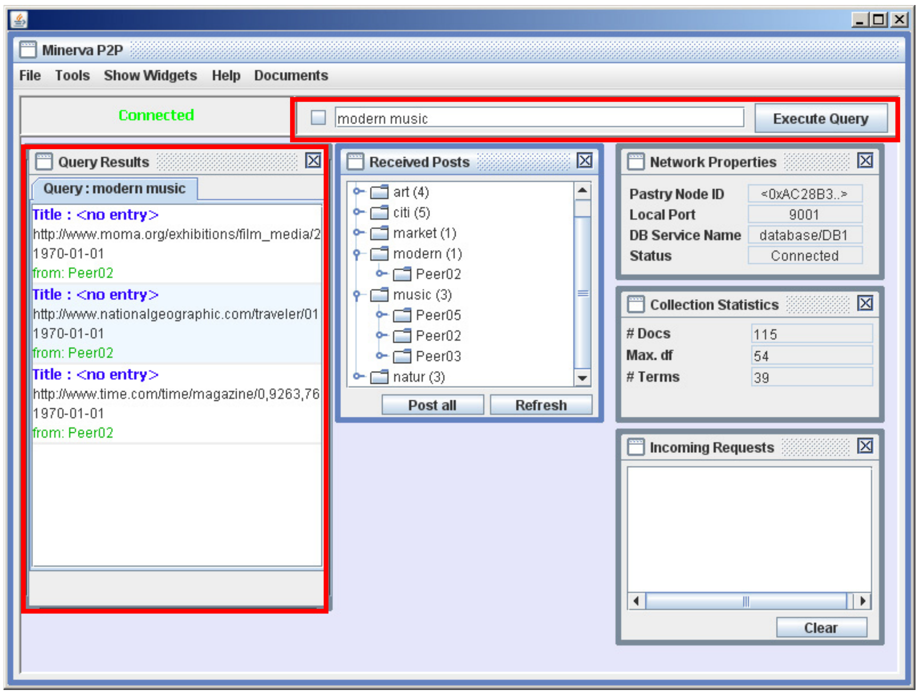
# Docs	115
Max. df	54
# Terms	39

Figure 6.5: Updating and Disseminating Metadata.

Peer selection for this query is trivial and the one-time query is forwarded to this peer. Nevertheless, Minerva allows to specify the peer selection strategy (e.g., CORI [SJCO02]) and the number of remote peers that receive the query. The *Query Results* widget shows the final result document list to the user. All three documents are hosted at *Peer02*. The URL allows to visit the document's online version. If more than one peer contribute their local results for the requested query, and if the overall number of results is high, Minerva merges the collected result documents (e.g., using CORI-based merging algorithms) and displays only the top-ranked documents to the user.

6.3.2.4 Continuous Query Execution

Subscribing with a continuous query using the Minerva prototype with extended MAPS functionality is shown in Figure 6.7. The user enters the continuous query in the text field mentioned before and selects the check box on the left to specify that this request is a continuous query. In the background, Minerva checks whether this continuous query is already *active*, i.e., it was requested by the same user in the past. If the query is active, the directory is used to retrieve updated statistical metadata about the query keys, and time series analysis is applied according to the MAPS approach presented in this thesis. Prediction methods such as double exponential smoothing can not be applied to continuous queries requested for the first time. Thus, *resource selection* alone is used to select the most promising publishers in the future. Having selected the interesting publisher peers, the Minerva prototype system sends the continuous query to them and waits for new published documents. Periodically, continuous queries have to be updated to recognize publishing behavior. This can be realized automatically by the system or manually by the user.



modern music

Title : <no entry>
http://www.moma.org/exhibitions/film_media/2
1970-01-01
from: Peer02

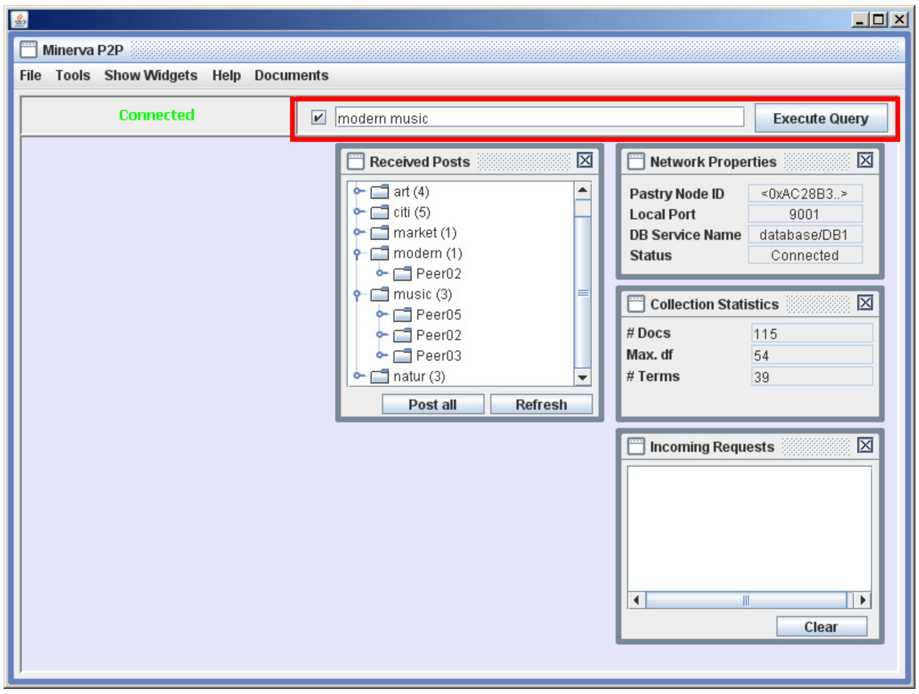
Title : <no entry>
http://www.nationalgeographic.com/traveler/01
1970-01-01
from: Peer02

Title : <no entry>
http://www.time.com/time/magazine/0,9263,76
1970-01-01
from: Peer02

Pastry Node ID	<0xAC28B3..>
Local Port	9001
DB Service Name	database/DB1
Status	Connected

# Docs	115
Max. df	54
# Terms	39

Figure 6.6: One-Time Query Execution with Minerva.

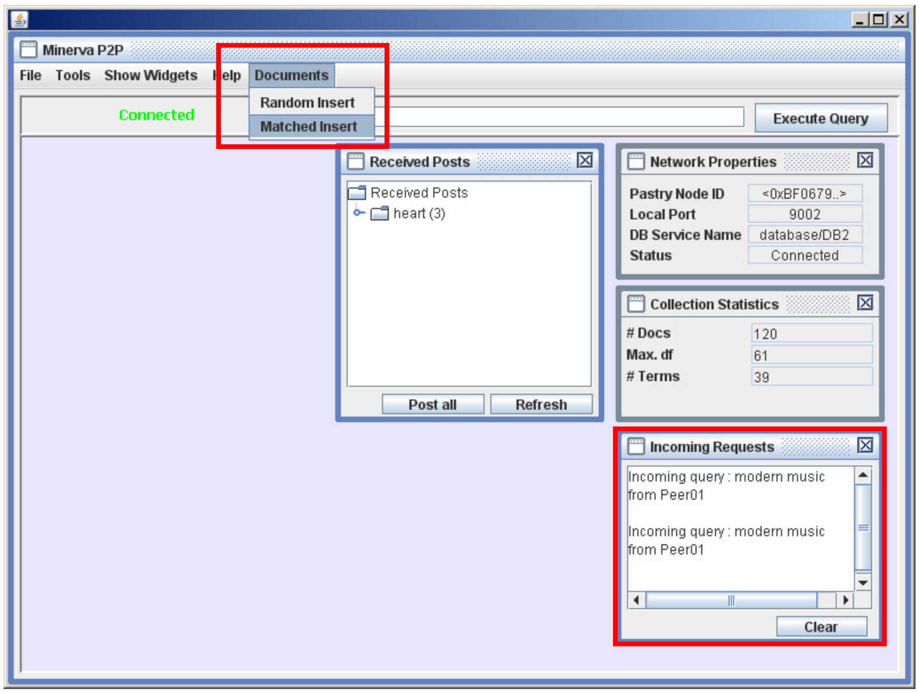


☑ modern music

Pastry Node ID	<0xAC28B3..>
Local Port	9001
DB Service Name	database/DB1
Status	Connected

# Docs	115
Max. df	54
# Terms	39

Figure 6.7: Continuous Query Execution with Minerva.



Pastry Node ID	<0xBF0679..>
Local Port	9002
DB Service Name	database/DB2
Status	Connected

# Docs	120
Max. df	61
# Terms	39

```
Incoming query : modern music
from Peer01

Incoming query : modern music
from Peer01
```

Figure 6.8: Publishing Documents with Notifying Subscriber Peers.

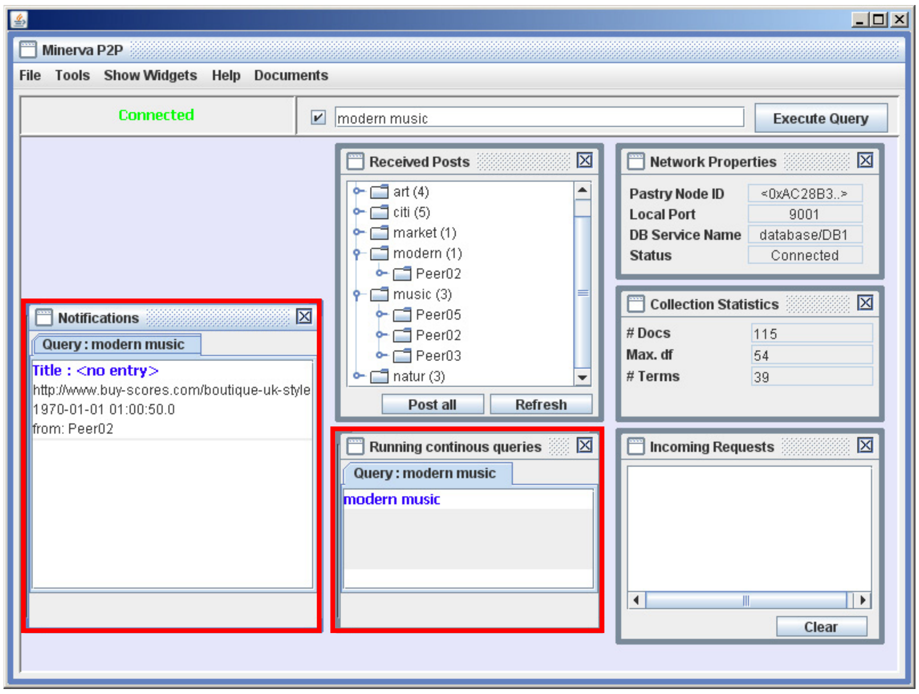
6.3.2.5 Document Publication

Figure 6.8 illustrates the publication process for the showcase where a peer can publish new documents by adding them to its own local collection. Each peer in the network has a database connection to this server, and two different methods allow to publish new documents:

- *Random Insert* selects a completely random document from the server collection and adds the selected one to the local database. A random document can not be published twice such that the server reminds of this publication.

- *Matched Insert* allows in this showcase to select documents that match active continuous queries a peer stores. In this case, the peer selects a random stored information demand and gets a matching document from the server to add it to its local store. If there is no matching document available or no active continuous query stored, a random publication occurs.

On the lower right, Figure 6.8 shows the incoming requests. The first request was the one-time query for *modern music* and the second request was the same query as long-term demand. Having added new documents to the local collection, a peer updates its *Collection Statistics* and disseminates its refreshed metadata to the network by pushing again the *Post all* button. This procedure can be done periodically to decrease network traffic caused by update messages.



Title : <no entry>
http://www.buy-scores.com/boutique-uk-style
1970-01-01 01:00:50.0
from: Peer02

Figure 6.9: Receiving Notifications for Subscribed Continuous Queries.

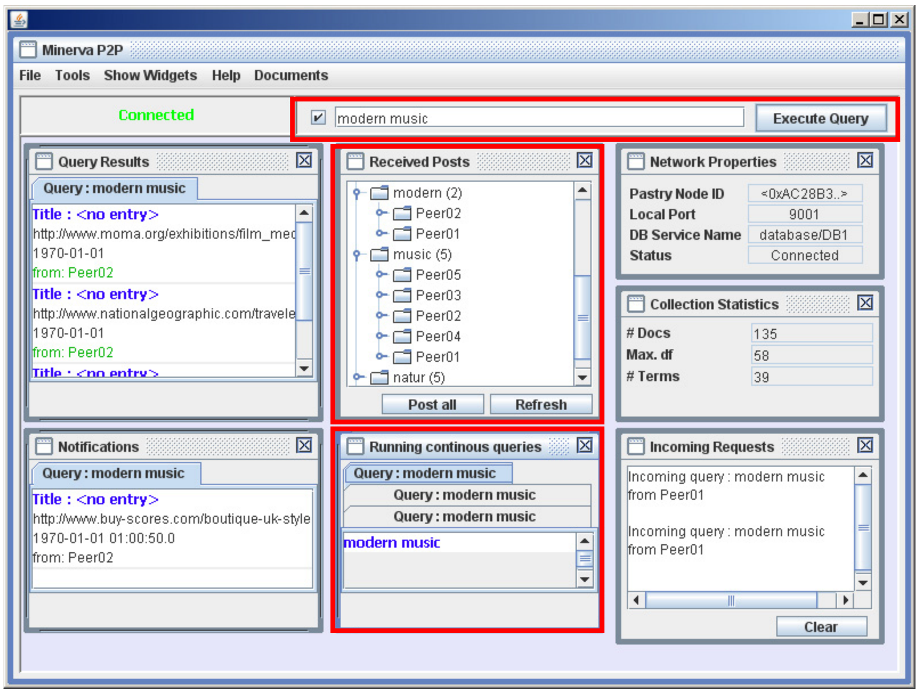
6.3.2.6 Receiving Notifications

Receiving a notification for a new published document is shown in Figure 6.9. Here, the querying peer for *modern music* gets a notification message from *Peer02*. This message is included in the *Notifications* widget. Of course, the user can directly follow the URL link to access the online-version of the published document. In addition, the *Running continuous queries* widget lists all active subscriptions of the current peer. Again, continuous queries with expired lifetime are removed from the list.

6.3.2.7 Resubmitting Continuous Queries

The last screenshot deals with the resubmission of already existing continuous queries. Figure 6.10 shows that the query *modern music* is requested again. In the meantime, several peers in the network have published new documents. The *Received Posts* widget lists all currently available metadata for the two requested keys. Now, two peers have published documents concerning *modern*, and five peers store data regarding *music*. Thus, peer selection for the whole query selects *Peer01* and *Peer02* also considering the time series observations since the query was last requested.

The *Running continuous queries* widget designates that this query was stored before, such that metadata of previous executions and selection processes are available to be used in the future.



Connected

modern music

Query : modern music
Title : <no entry>
http://www.moma.org/exhibitions/film_med
1970-01-01
from: Peer02
Title : <no entry>
http://www.nationalgeographic.com/travele
1970-01-01
from: Peer02
Title : <no entry>

modern (2)
Peer02
Peer01
music (5)
Peer05
Peer03
Peer02
Peer04
Peer01
natur (5)

Pastry Node ID	<0xAC28B3..>
Local Port	9001
DB Service Name	database/DB1
Status	Connected

# Docs	135
Max. df	58
# Terms	39

Query : modern music
Title : <no entry>
http://www.buy-scores.com/boutique-uk-style
1970-01-01 01:00:50.0
from: Peer02

Query : modern music
Query : modern music
Query : modern music
modern music

Incoming query : modern music
from Peer01

Incoming query : modern music
from Peer01

Figure 6.10: Resubmitting a Continuous Query.

6.4 Other Prototypes

This sections briefly introduces some existing prototypes for P2P retrieval and P2P filtering. *LibraRing* [TIK05a] is the only other system that combines retrieval and filtering functionality in a P2P environment of digital libraries. In contrast to the prototype presented in this chapter, LibraRing focuses on exact searching and filtering functionality by disseminating documents or continuous queries in the network. Section 6.4.1 presents some P2P retrieval systems, whereas Section 6.4.2 discusses relevant P2P filtering systems. Overall, the prototype presented in this thesis is the only approach that provides approximate P2P searching and filtering functionality in a unifying framework.

6.4.1 P2P Retrieval Prototypes

Galanx [WGD03, GWJD03] is a P2P search engine implemented using the *Apache* HTTP server and *BerkeleyDB*. Web site servers form the P2P layer of this architecture; pages are stored only where they originate from. Galanx directs user queries to relevant peers by consulting a local peer index that is maintained on each peer. In the experimental evaluation, the use of peer indices to direct searches is investigated. In contrast, the Minerva approach relies on peers to decide at what extent they want to crawl interesting fractions of the Web and build their own local indexes. [GWJD03] focuses on *XML* data repositories and postulates that, upon completion of the query, regardless of the number of results or how they are ranked and presented, the system guarantees that all the relevant data sources known at query submission time have been contacted. For this purpose, a distributed catalog service that maintains summaries of all peers is designed.

PlanetP [CAPMN03] is a publish-subscribe service for unstructured P2P communities, supporting content ranking search. PlanetP distinguishes local indexes and a global index to describe all peers and their shared information. The global index is replicated using a gossiping algorithm. PlanetP does not provide notification messages about new published data. *Odissea* [SMwW+03] (Open DIStributed Search Engine Architecture) assumes a two-layered search engine architecture with a global index structure distributed over the peers in the system. The system provides a highly distributed global indexing and query execution service that can be used for content residing inside or outside the P2P network. A single peer holds the complete, Web-scale, index for a given text key (i.e., keyword or word stem). Query execution uses a distributed version of *Fagin's threshold algorithm* [Fag02]. The system appears to cause higher network traffic when posting document metadata into the network, and the presented query execution method seems limited to queries with at most two keywords. The paper actually advocates using a limited number of peers, in the spirit of a server farm.

The *OverCite* system [SCL+05] was proposed as a distributed alternative for the scientific literature digital library *CiteSeer*. This functionality was made possible by utilizing a DHT infrastructure to harness distributed resources (storage, computational power, etc.). OverCite is able to support new features such as documents alerts. The work presented in [RV03] adopts an architecture very similar to Minerva, but seems incomplete since one cannot locate a running implementation. The presented results are based on simulations that also support the assumption that a Minerva-like architectures do in fact scale and are well within reasonable bandwidth limits. The system described in the paper provides keyword search functionality for a DHT-based file system or archival storage system, to map keyword queries to unique routing keys. It does so by mapping each keyword to a peer in the DHT that will store a list of documents containing that keyword.

The *eSearch* system presented in [TD04] is a P2P keyword search system based on a hybrid indexing structure in which each peer is responsible for certain keys. Given a document, eSearch selects a small number of important keys in the document and publishes the complete key list for the document to peers responsible for those top keys. This selective replication of key lists allows a multi-key query to be processed locally at the peers responsible for the query keys, but the document granularity indexes may interfere with the goal of unlimited scalability. The authors claim that eSearch is scalable and efficient, and obtains search results as good as state-of-the-art centralized systems.

Rumorama [EH05] is an approach based on the replication of peer data summaries via rumor spreading and multi-casting techniques in a structured overlay. Rumorama utilizes a hierarchical structure, and adopts a summary-based approach to support P2P-IR in the spirit of *PlanetP*. In a Rumorama network, each peer views the network as a small PlanetP network with connections to peers that see other small PlanetP networks. Each peer can select the size of the PlanetP network it wants to see according to its local processing power and bandwidth. Rumorama manages to process a query such that the summary of each peer is considered exactly once in a network without churn. The actual number of peers to be contacted for a query is a small fraction of the total number of peers in the network.

Alvis [LKP+05] is a prototype for scalable full-text P2P-IR using the notion of *Highly Discriminative Keys* (*HDK*) for indexing, which claims to overcome the scalability problem of single-key retrieval in structured P2P networks. Alvis is a fully-functional retrieval engine built on top of *P-Grid*. It provides distributed indexing, retrieval, and a content-based ranking module. While the index size is even larger than the single key index, the authors bring forward that storage is available in P2P systems as opposed to network bandwidth. ALVIS includes a component for HDK-based indexing and retrieval, and a distributed content-based ranking module.

6.4.2 P2P Filtering Prototypes

Scribe [RKCD01] is a large-scale event notification infrastructure for topic-based publish-subscribe applications. It supports large numbers of topics, with a potentially large number of subscribers per topic. Scribe is built on top of Pastry [RD01a], and leverages Pastry's reliability, self-organization and locality properties. Pastry is used to create a topic (group) and to build an efficient multi-cast tree for the dissemination of events to the topic's subscribers. *Hermes* [FFS$^+$01] is similar to Scribe because it uses the same underlying DHT but it allows more expressive subscriptions by supporting the notion of an event type with attributes. Hermes offers sophisticated filtering capabilities preventing the user from notifications about non-interesting events. From the user's point of view Hermes integrates the providers into a single source. Its simple provider interface makes it easy for publishers to join the service and thus reaching the potential readers directly.

The *pFilter* system [TX03] is a global-scale decentralized information filtering and dissemination system for unstructured documents that connects potentially millions of computers in national (and international) computing Grids or ordinary desktops into a structured P2P overlay network. The pFilter system uses a hierarchical extension of the CAN DHT to filter unstructured documents and relies on multi-cast trees to notify subscribers. VSM and LSI can be used to match documents to user queries. The *DHTStrings* system [AT05] utilizes a DHT-agnostic architecture to develop algorithms for efficient multi-dimensional event processing. It addresses the issue of supporting efficiently queries over string-attributes involving prefix, suffix, containment, and equality operators in large-scale data networks.

PeerCQ is a totally decentralized system that performs information monitoring tasks over a network of peers with heterogeneous capabilities. It uses Continual Queries (CQs) as its primitives to express information monitoring requests. A primary objective of the PeerCQ system is to build a decentralized Internet-scale distributed information monitoring system, which is highly scalable, self-configurable and supports efficient and robust CQ processing. PeerCQ's most important contribution is that it takes into account peer heterogeneity and extends consistent hashing with simple load-balancing techniques based on appropriate assignment of peer identifiers to network peers.

P2P-DIET [IKT04a] utilizes an expressive query language based on IR concepts and is implemented as an unstructured P2P network with routing techniques based on shortest paths and minimum weight spanning trees. P2P-DIET has been implemented on top of the open source *DIET Agents Platform*[4] [HWBM02] and combines ad-hoc querying as found in other super-peer networks and also as proposed in the DIAS system [KKTR02]. Finally, the *AGILE* system [DFK05] presents Context-aware Information Filters (*CIF*) using two input streams with messages and context updates. The system extends existing index structures such that the indexes adapt to message/update workload with satisfying performance results.

6.5 Discussion

This chapter presented the current prototype implementation of the MAPS approximate information filtering approach. The prototype extends the Minerva search system with additional components to realize filtering functionality. An extensive showcase illustrated the usage of the extended prototype system. The following chapter will present a use case for digital libraries that supports retrieval and filtering functionality under a single unifying framework.

[4]http://diet-agents.sourceforge.net/

Chapter 7

Digital Library Use Case

This chapter presents a *digital library* use case using the *MinervaDL* architecture. MinervaDL is build upon a two-tier version of the MAPS architecture and designed to support approximate retrieval and filtering functionality under a single unifying framework. The architecture of MinervaDL as introduced in [ZTW07] is able to handle huge amounts of data provided by digital libraries in a distributed and self-organizing manner. The super-peer architecture and the use of the *distributed hash table* as the routing substrate provides an infrastructure for creating large networks of digital libraries with minimal administration costs.

Section 7.1 introduces the main characteristics of this DL use case, and discusses some related work in this area. The high-level DL architecture including the involved components is presented in Section 7.2, whereas Section 7.3 explains in detail the appropriate protocols to ensure the two functionalities (retrieval and filtering). Section 7.4 discusses the two different scoring functions to ensure approximate search and filtering, and Section 7.5 presents the experimental evaluation of MinervaDL. Finally, Section 7.6 concludes this chapter.

7.1 Introduction

This chapter presents a novel DL architecture called MinervaDL; it is designed to support *approximate* information retrieval and filtering functionality in a single super-peer-based architecture. In contrast to the MAPS architecture presented in the previous chapters, MinervaDL is hierarchical like the ones in [TIK05a, LC03, SMwW$^+$03, TZWK07, RPTW08] and utilizes a DHT to achieve scalability, fault-tolerance, and robustness in its routing layer. The MinervaDL architecture allows handling huge amounts of data provided by DLs in a distributed and self-organizing way, and provides an infrastructure for creating large networks of digital libraries with minimal administration costs. There are two kinds of basic functionality that are offered in MinervaDL:

- *Information Retrieval*: In an information retrieval scenario (also known as *one-time querying*), a user poses an *one-time query* and the system returns all resources matching the query (e.g., all currently available documents relevant to the requested query).

- *Information Filtering*: In an information filtering scenario (also known as *publish/subscribe* or *continuous querying* or *selective dissemination* of information), a user submits a *continuous query* (or *subscription* or *profile*) and will later be notified from the system about certain events of interest that take place (i.e., about newly published documents relevant to the continuous query).

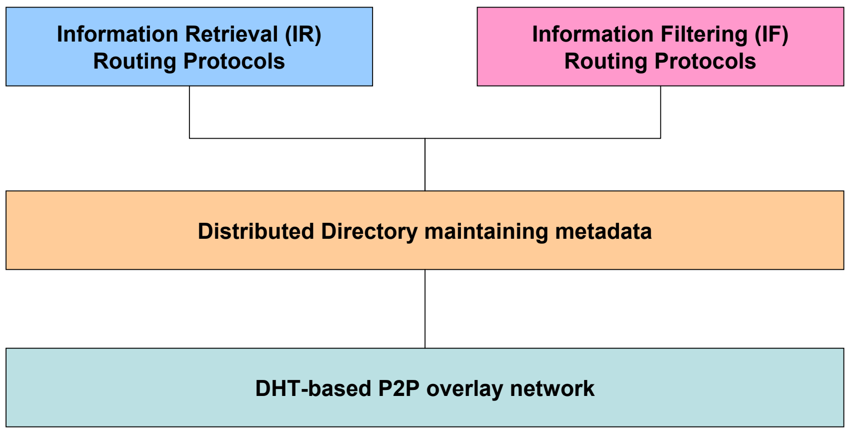

Information Retrieval (IR) Routing Protocols
Information Filtering (IF) Routing Protocols
Distributed Directory maintaining metadata
DHT-based P2P overlay network

Figure 7.1: DHT-based Distributed Directory to Perform IR and IF.

The proposed DL architecture is built upon a distributed directory similar to [BMT+05b, BMT+05a] that stores metadata. Routing protocols for information filtering and retrieval use this directory information to perform the two functionalities. Figure 7.1 shows this design principle. MinervaDL identifies three main components: *super-peers*, *providers*, and *consumers*. Providers are implemented by information sources (e.g., digital libraries) that want to expose their content to the rest of the MinervaDL network, while consumers are utilized by users to query for and subscribe to new content. Super-peers utilize the *Chord* DHT [SMLN+03] to create a conceptually global, but physically distributed directory that manages aggregated statistical information about each provider's local knowledge in compact form. This distributed directory allows information consumers to collect statistics about information sources and rank them according to the probability to answer a specific information need. This reduces network costs and enhances scalability since only the most relevant information sources are queried. In MinervaDL, both publications and (one-time and continuous) queries are interpreted using the *vector space model* (VSM), but other appropriate data models and languages could also be used, e.g., *latent semantic indexing* (LSI) or language models.

7.1.1 A Motivating Example

To give an better understanding of the potential benefits of a system that integrates both IR and IF functionality in the DL context, consider the example of John, a professor in computer science, who is interested in *constraint programming*. He wants to follow the work of prominent researchers in this area. He regularly uses the digital library of his department and a handful of other digital libraries to search for new papers in the area. Even though searching for interesting papers this week turned up nothing, a search next week may turn up new information. Clearly, John would benefit from accessing a system that is able to not only provide a search functionality that integrates a big number of sources (e.g., organizational digital libraries or even libraries from big publishing houses), but also capture his long term information need (e.g., in the spirit of [TIK05a, PB02, YJ06]).

This system would be a valuable tool, beyond anything supported in current digital library systems, that would allow John to save time and effort. In the example scenario, the university John works in is comprised of three geographically distributed campuses (Literature, Sciences, and Biology) and each campus has its own local digital library. In the context of MinervaDL, each campus would maintain its own super-peer, which provides an access point for the provider representing the campus' digital library, and the clients deployed by users such as John. Other super-peers may also be deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer, or Elsevier), to provide access points for their users (students, faculty or employees) and make the contents of their digital libraries available in a timely way. MinervaDL offers an infrastructure, based on concepts of P2P systems, for organizing the super-peers in a scalable, efficient and self-organizing architecture. This architecture allows seamless integration of information sources, enhances fault-tolerance, and requires minimum administration costs.

7.1.2 The Evolution of Digital Libraries

In [GT02], a digital library is defined as follows: *A digital library is a library in which collections are stored in digital formats (as opposed to print, microform, or other media) and accessible by computers. The digital content may be stored locally, or accessed remotely via computer networks.* The term *digital library* was used the first time in 1988 in a report to the Corporation for National Research Initiatives. The older names *electronic library* or *virtual library* are also occasionally used, though electronic library nowadays more often refers to portals, often provided by government or public agencies (e.g., the *Florida Electronic Library*). Digital libraries (DLs) have been made possible, due to the integration and the use of a number of information technologies, the availability of digital content on a global scale, and a strong demand for users who are now online. They are destined to become an essential part of the information infrastructure in the 21st century (*Information Age* also known as *Digital Age* or *Wireless Age*).

The term digital library can be applied to a wide range of collections and organizations, but, to be considered a digital library, an online collection of information must be managed by and made accessible to a community of users. A lot of known digital libraries are older than the Web (e.g., *Project Gutenberg*). But, as a result of the development of the Web and its search potential, digital libraries are now moving towards Web-based environments. Often, there is a distinction between content (created in a digital format), known as born-digital, and information (converted from a physical medium), e.g., paper, by digitizing.

Most digital libraries provide a search interface to locate resources (typically *Deep Web* resources that cannot be located by Web search engines). Some digital libraries create special pages or sitemaps to allow search engines to find all their resources. Digital libraries frequently use the *Open Archives Initiative Protocol for Metadata Harvesting* (OAI-PMH) [LdS01] to expose their metadata to other digital libraries, and search engines like *Google*, and *Yahoo*. There are two general strategies for searching a *federation of digital libraries*: (i) distributed searching, and (ii) searching previously harvested metadata.

Distributed searching typically involves a client sending multiple search requests in parallel to a number of servers in the federation. The results are gathered, duplicates are eliminated or clustered, and the remaining items are sorted and presented back to the client. Protocols like *Z39.50* are frequently used in distributed searching. A benefit to this approach is that the resource-intensive tasks of indexing and storage are left to the respective servers in the federation. A drawback to this approach is that the search mechanism is limited by the different indexing and ranking capabilities of each database, making it difficult to assemble a combined result consisting of the most relevant found items.

Searching over previously harvested metadata involves searching a locally stored index of information that has previously been collected from the libraries in the federation. When a search is performed, the search mechanism does not need to make connections with the digital libraries it is searching - it already has a local representation of the information. This approach requires the creation of an indexing and harvesting mechanism which operates regularly; it connects to all the digital libraries and queries the whole collection in order to discover new and updated resources. *OAI-PMH* is frequently used by digital libraries for allowing metadata to be harvested. A benefit to this approach is that the search mechanism has full control over indexing and ranking algorithms, possibly allowing more consistent results. A drawback is that harvesting and indexing systems are more resource-intensive and therefore expensive. Now, large scale digitization projects are underway (e.g., *Google Books*). With continued improvements in book handling and presentation technologies, and development of alternative depositories and business models, digital libraries are rapidly growing in popularity. Beyond that, libraries have just ventured into audio and video collections (e.g., digital libraries such as the *Internet Archive*).

7.1.3 Previous Research on P2P Digital Library Architectures

P2P-DIET [IKT04a] and *LibraRing* where the first approaches that tried to support both IR and IF functionalities in a single unifying framework. P2P-DIET utilizes an expressive query language based on IR concepts and is implemented as an *unstructured P2P network* with routing techniques based on shortest paths and minimum weight spanning trees. An extension of P2P-DIET [CIKN04] considers a similar problem for distributing RDF metadata in an *Edutella* [NWQ+02] fashion. LibraRing [TIK05a] was the first approach to provide protocols for the support of both IR and IF functionality in DLs using DHTs. In LibraRing, super-peers are organized in a Chord DHT and both (continuous) queries and documents are indexed by hashing words contained in them. This hashing scheme depends heavily on the data model and query language adopted, and the protocols have to be modified when the data model changes [TIK05a]. The DHT is used to make sure that queries meet the matching documents (in the IR scenario) or that published documents meet the indexed continuous queries (in the IF scenario). In this way the retrieval effectiveness of a centralized system is achieved, while a number of routing optimizations (such as value proxying, content based-multicasting, etc.) are used to enhance scalability. [RPTW08] presents *iClusterDL*, a self-organizing overlay network that supports information retrieval and filtering functionality in a digital library environment. Contrary to approaches like LibraRing [TIK05a] that focus on *exact* retrieval and filtering functionality (e.g., by disseminating documents or continuous queries in the network), in *MinervaDL* publications are processed locally and query or subscribe to only selected information sources that are most likely to satisfy the user's information demand. In this way, efficiency and scalability are enhanced by trading faster response times for some loss in recall, achieving *approximate* retrieval and filtering functionality. MinervaDL is the first approach to provide a comprehensive architecture and the related protocols to support approximate retrieval and filtering functionality in a digital library context. Contrary to the LibraRing approach, in MinervaDL the Chord DHT is used to disseminate and store metadata about the document providers rather than the documents themselves. Avoiding per-document indexing granularity allows to improve scalability by trading recall for lower message traffic. This approximate retrieval and filtering approach relaxes the assumption of potentially delivering notifications from every producer that holds in the works mentioned above and amplifies scalability. Additionally, it allows to easily support different data models and query languages, without modifications to the protocols, since matching is performed locally in each peer.

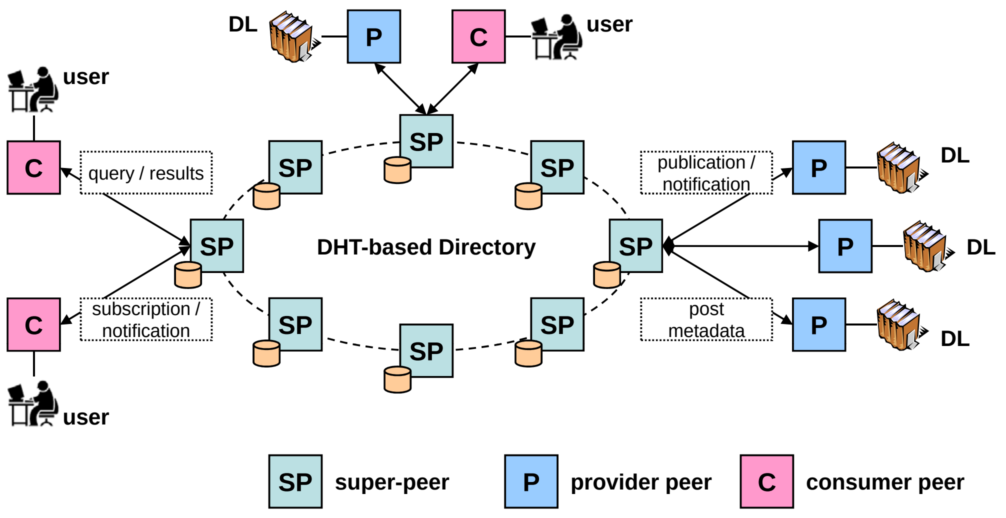


Figure 7.2: High-Level View of the MinervaDL Architecture.

7.2 The MinervaDL Architecture

This section of the use case presents the high-level view of the MinervaDL architecture and presents the various system components. The system architecture of MinervaDL is composed of three different types of peers: *super-peers*, *consumer peers* (or *consumers*), and *provider peers* (or *providers*).

Figure 7.2 shows a high-level view using an underlying DHT-based directory (e.g., Chord [SMK+01]). The following sections explain the three main components and explain their properties and abilities in detail while Section 7.3 presents the protocols regulating peer interactions.

7.2.1 Super-Peers

Super-peers run the DHT protocol and form a distributed directory that maintains statistics (metadata) about providers' local knowledge in compact form. In MinervaDL, the Chord DHT is used to partition the key space such that each directory peer (super-peer) is responsible for the statistics of a randomized subset of keys. Directory peers are super-peers, peers with more capabilities than consumer or provider peers (e.g., more cpu power and bandwidth capacities) that are responsible for serving information consumers and providers and act as their *access point* to the MinervaDL network. When the number of super-peers is small, each peer can easily locate others in a single hop by maintaining a full routing table. When the super-peer network grows in size, the DHT provides a scalable means of locating other super-peers in the network.

Super-peers can be deployed by large institutions like universities, research centers or content providers (e.g., CiteSeer, ACM, Springer, Elsevier) to provide access points for their users (students, faculty or employees) or digital libraries. As shown in Figure 7.2, more than one provider and/or consumer can be connected to a single super-peer that acts as their common access point.

7.2.2 Consumer Peers

Consumer peers (or *consumers*) are utilized by users (e.g., students, faculty or employees) to connect to the MinervaDL network, using a single super-peer as their access point. Utilizing a consumer peer allows users to pose one-time queries, receive relevant resources, subscribe to resource publications with continuous queries and receive notifications about published resources (e.g., documents) that match their interests. Consumer peers are responsible for selecting the best information sources to query (respectively monitor) with respect to a given one-time query (respectively continuous query). If consumer peers are not online to receive notifications about documents matching their submitted continuous queries, these notifications are stored by their access point and are delivered upon reconnection. Section 7.3 presents the protocols regulating the activities of consumer peers.

7.2.3 Provider Peer

Provider peers (or *providers*) are implemented by information sources that want to expose their content to the MinervaDL network. Typical examples are digital libraries deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer, or Elsevier). Provider peers use a directory peer (super-peer) as their access point and utilize it to distribute statistics about their local resources to the network. Providers answer one-time queries and store continuous queries submitted by consumers to match them against new documents they publish. More than one provider peers may be used to expose the contents of large digital libraries, and also an integration layer can be used to unify different types of DLs.

7.3 The MinervaDL Protocols

Having introduced in the previous section the main architecture of MinervaDL with three different types of peers, in this section, the protocols that regulate the interactions between all types of peers in the DL architecture are explained in detail. The protocols include the joining and leaving of consumers, providers, and super-peers, but also the publication of new documents, the submission of one-time or continuous queries, and the receipt of answers and notifications for submitted requests.

Before explaining the individual protocols, three different functions have to be defined. These will be used to ensure the basic communication procedures of the presented protocols:

- Function $key(n)$ denotes the *key* of a consumer or provider peer n. The key is created the first time a consumer or provider joins the MinervaDL network by using, e.g., the IP address, port, and timestamp of the joining peer. It is used to support dynamic IP addressing.
- The *identifier* of a peer n is denoted by function $id(n)$. This identifier is produced by applying the Chord hash function and is used to identify a super-peer within the Chord ring.
- Function $ip(n)$ refers to the *IP address* of a peer n, and is applied to get the current contact address of that peer. Whereas function $key(n)$ is only applied to consumer or provider peers, and function $id(n)$ only to super-peers, all three peer types need an IP address resolved by function $ip(n)$.

7.3.1 Provider & Consumer Join/Leave

The *first time*, a provider peer P wants to connect to the existing MinervaDL network, it has to follow the join protocol. P has to find the IP address of a super-peer S using out-of-band means (e.g., via a secure Web site that contains IP addresses for the super-peers that are currently online in the network). Then, P sends to S a NewProv($key(P), ip(P)$) message, and S adds P in its local *provider table* (PT), which is a hash table used for identifying the providers that use S as their *access point*. Here, $key(P)$ is used to index providers in PT, while each PT slot stores contact information about the provider including its status (connected or disconnected) and its stored notifications (see Section 7.3.8 for notification delivery).

Subsequently, super-peer S sends to provider P an appropriate acknowledgement message AckNewProv($id(S), ip(S)$). Once P has joined, it can use the connect/disconnect protocol described next to connect to and disconnect from the network. A consumers C uses a similar protocol to join the MinervaDL network. In this case the appropriate messages NewCons and AckNewCons are utilized in combination with a *consumer table* CT managing contact information about consumers.

A provider peer P or a consumer peer C that want to leave the network has to send a LeaveProv($key(P), ip(P), id(S)$) or LeaveCons($key(C), ip(C), id(S)$) message to its access point S, respectively. The super-peer S deletes the peer from its provider or consumer table including all contact information.

7.3.2 Provider & Consumer Connect/Disconnect

When a provider P wants to connect to the network, it sends to its access point S a ConnectProv($key(P), ip(P), id(S)$) message. If $key(P)$ exists in PT of S, P is marked as connected. If $key(P)$ does not exist in PT, this means that S was not the access point of P the last time that P connected (Section 7.3.8 discusses this case). When a provider P wants to disconnect, it sends to its access point S a DisconnectProv($key(P), ip(P), id(S)$) message and S marks P as disconnected in its PT.

Consumers connect/disconnect from the network in a similar way (applying messages ConnectCons and DisconnectCons), but S has also to make sure that a disconnecting consumer C will not miss notifications about resources of interest while not online. Thus, notifications for C are stored in the *consumer table* CT of S and wait to be delivered upon reconnection of C (see Section 7.3.8).

7.3.3 Super-Peer Join/Leave

To join the MinervaDL network, a super-peer S must find the IP address of another super-peer S' using out-of-band means. S creates a NewSPeer($id(S), ip(S)$) message and sends it to S' which performs a lookup operation by calling *lookup(id(S))* to find $S_{succ} = successor(id(S))$, similarly to the Chord joining procedure. S' sends a AckNewSPeer($id(S_{succ}), ip(S_{succ})$) message to S and S updates its successor to S_{succ}. S also contacts S_{succ} asking its predecessor and the data that should now be stored at S. S_{succ} updates its predecessor to S, and answers back with the contact information of its previous predecessor, S_{pred}, and all continuous queries and publications that were indexed under key k, with $id(S) \leq k < id(S_{pred})$. S makes S_{pred} its predecessor and populates its index structures with the new data that arrived. After that S populates its finger table entries by repeatedly performing lookup operations on the desired keys.

When a super-peer S wants to leave MinervaDL network, it constructs a DISCONNECT-SPEER$(id(S), ip(S), id(S_{pred}), ip(S_{pred}), data)$ message, where $data$ are all the continuous queries, published resources and stored notifications of off-line peers that S was responsible for. Subsequently, S sends the message to its successor S_{succ} and notifies S_{pred} that its successor is now S_{succ}. Clients that used S as their access point connect to the network through another super-peer S'. Stored notifications can be retrieved through $successor(id(S))$.

7.3.4 Directory Maintenance

In MinervaDL, the super-peers utilize a Chord-like DHT [SMK+01] to build-up a distributed directory, while each provider P uses its access point to distribute per-key statistics about its local index to the directory using POST messages. At certain intervals (e.g., every t publications or time units) the provider has to update its statistics in the directory. Now, the updating process done by a provider P is described.

Let $K = k_1, k_2, \ldots, k_n$ denote the set of all keys included in the documents a provider P has published after the last directory update. For each key $k \in K$, the provider computes statistics: (i) the maximum term frequency of occurrence within P's document collection (tf_k^{max}); (ii) the number of documents in its document collection containing t (df_k); (iii) the size of P's document collection (cs). Using its IP address, P forwards the POST$(key(P), ip(P), tf_k^{max}, df_k, cs, k)$ message to the super-peer S that is P's access point to the directory.

Next, S uses the Chord *lookup()* function to forward a modified POST message (including in addition S's IP address $ip(S)$ and identifier $id(S)$) to the super-peer responsible for identifier $H(k)$ (i.e., this peer is responsible for maintaining statistics for key k). This peer S_k stores the received POST message in its local *statistics table* ST to be able to provide the key statistics to peers requesting them.

7.3.5 Submitting an One-Time Query

In this section it is shown how to answer one-time vector space queries. Let us assume that a consumer C wants to submit a one-time query q containing keys $k_1, k_2, \ldots, k_n$. The following steps take place to execute this query:

In step one, consumer C sends to its access point S a SUBMITQ$(key(C), ip(C), q)$ message. This message includes besides the query also C's contact information necessary to receive answering messages.

In the second step, for each key $k \in q$, S computes $H(k)$ to obtain the identifier of the super-peer responsible for storing statistics about key k. Then, it sends a GET-STATS$(key(C), ip(C), k)$ message by using the Chord *lookup()* function.

In step three each super-peer S_k that receives a GETSTATS message, searches its local statistics table ST for key k to retrieve a list L of provider peers storing documents containing the key t. Each element in list L is a tuple $(key(P), ip(P), tf_k^{max}, df_k, cs, k)$ containing contact information about providers and statistics about keys contained in documents that these providers publish. Subsequently, S_k creates a RETSTATS$(id(S_k), ip(S_k), L)$ message and sends it to consumer C using $ip(C)$ included in the GETSTATS message. In this way, the consumer receives provider statistics for all query keys.

In step four, C uses the scoring function $sel(P, q)$ described in Section 7.4 to rank the providers with respect to q and identify the top-k providers that hold documents satisfying q. This scoring function is based on resource selection algorithms known from the IR literature (see Section 7.4).

Subsequently, C creates a GETRESULTS$(ip(C), key(C), q)$ message and forwards it, using the contact information associated with the statistics, to all provider peers selected previously. Once a provider peer P receives a GETRESULTS message containing a query q, it matches q against its local document collection to retrieve the documents matching q. The local results are ranked according to their relevance to the query to create a result list R. Subsequently, P creates a RETRESULTS$(ip(P), R, q)$ message and sends it to C. In this way, C collects the local result lists of all selected providers and uses them to compute a final result list that is then presented to the user. To merge the retrieved result lists, standard IR scoring functions (e.g., CORI [CLC95], GlOSS [GGMT99], or CVV [YL97]) are used. In [FPC+99], various standard approaches are compared.

7.3.6 Subscribing with a Continuous Query

This section describes how to extend the protocols of Section 7.3.5 to provide information filtering functionality. To submit a continuous query cq containing keys $k_1, k_2, \ldots, k_n$, the one-time query submission protocol needs to be *modified*. The first three steps are identical while step four is modified as follows.

C uses the scoring function $pred(P, cq)$ described in Section 7.4 to rank the providers with respect to cq and identify the $top-k$ providers that may publish documents matching cq *in the future*. These are the peers that will store cq and C will receive notifications from these peers only. This query indexing scheme makes provider selection a critical component of the filtering functionality. Notice that, in a filtering setting, resource selection techniques like $sel(P, cq)$ described in Section 7.4 and used for one-time querying, are not appropriate since MinervaDL is not interested in the current document collection of the providers but rather in their future publishing behavior.

Once providers that will store cq have been determined, consumer C creates an message INDEXQUERY$(key(C), ip(C), id(S), ip(S), cq)$ and sends it to these providers using the IP addresses associated with the GETSTATS messages C received in the previous step. When a provider peer P receives an INDEXQUERY message, it stores cq in its local continuous query data structures to match it against future publications. P utilizes these data structures at publication time to find quickly all continuous queries that match a publication. This can be done using efficient algorithms, e.g., *BestFitTrie* [TKD04], or SQI [YGM99].

7.3.7 Publishing a new Document

Contrary to approaches such as [TIK05a] that distribute the documents or the index lists among the peers to achieve exact retrieval and filtering functionality, publications in MinervaDL are kept locally at each provider. This lack of publication forwarding mechanism is a design decision that offers increased scalability in MinervaDL by trading recall. Thus, only monitored provider peers (i.e., indexing a continuous query cq) can notify a consumer C, although other providers may also publish relevant documents. As already stated, this makes the scoring function $pred(P, cq)$ a critical component.

Thus, when a provider peer P wants to publish a new document d to the MinervaDL network, the document is only matched against P's local continuous query database to determine which continuous queries match d, and thus which consumers should be notified. Additionally, at certain intervals P creates POST messages with updated statistics and sends them to its access point S.

7.3.8 Notification Delivery

Assume a provider P that has to deliver a notification for a continuous query cq to consumer C. It creates a $\textsc{Notify}(ip(P), key(P), d, cq)$ message, where d is the document matching cq, and sends it to C. If C is not online at that time, then P sends the message to S, where S is the access point of C, using $ip(S)$ associated with cq. S then is responsible for storing the message and delivering it to C upon reconnection. If S is also off-line then the message is sent to S', which is the access point of P, and S' utilizes the DHT to locate the $successor(id(S))$ (as defined in Chord [SMLN+03]), by calling function *lookup()*. Answers to one-time queries are handled in a similar way.

7.4 Scoring Functions

Selecting the appropriate provider peers to forward an one-time or a continuous query requires a ranking or scoring function that will be used to determine the most appropriate sources (providers) for a given information demand. Although, both IR and IF protocols utilize the same metadata stored in the distributed directory, the peer ranking strategies in this use case differ significantly and have to consider varying objectives:

- In the case of information retrieval, the scoring algorithm has to identify *authorities* with respect to a given query q, i.e., peers that have already made available a lot of relevant document *in the past*. These peers should receive high scores, and thus be ranked high in the respective peer ranking that will result. For this purpose, standard resource selection algorithms known from the IR literature can be utilized. Section 7.4.1 discusses the scoring function for one-time querying.

- In contrast, in the information filtering case, the scoring function has to determine the most appropriate provider peers that given a continuous query cq, will publish documents matching *cq in the future*. In this setting, relying on existing resource selection algorithms similar to the ones utilized for one-time querying leads to low recall, as these algorithms are able to capture past publishing behavior, and cannot adapt to the dynamics of the filtering case.

So, it is clear, that both functionalities have to consider different objectives. In the filtering scenario, the main approach of this doctoral theses is used to select the best providers. To even better illustrate this, consider the following example: Assume two providers, where the first is specialized in *soccer* (i.e., in the past has published a lot of documents about soccer), although now it is rarely publishing new documents. The second provider is not specialized in soccer but currently it is publishing many documents about soccer. Now, a consumer subscribes beginning 2008 for documents with the continuous query *soccer Euro 2008*[1]. A ranking function based on *resource selection* algorithms would always choose the first provider. To get a higher ranking score, and thus get selected for indexing the query, the second provider has to specialize in soccer, a long procedure that is inapplicable in a filtering setting, which is by definition dynamic. The above example illustrates that the ranking formula should be able to predict the publishing behavior of providers by observing both their past and current behavior and projecting it to the future. Section 7.4.2 discusses the appropriate technique from this doctoral thesis.

[1]The 2008 UEFA European Football Championship, commonly referred to as *Euro 2008*, takes place in Austria and Switzerland, from 7 to 29 June 2008.

7.4.1 Resource Selection

As mentioned before, a number of *resource selection* methods (e.g., CORI [CLC95], GlOSS [GGMT99], CVV [YL97], language models, etc.) are available to identify good authorities with respect to a given multi-key query q. A simple but effective approach utilizes term frequencies (tf) and document frequencies (df) already stored in the directory to compute a score $sel(P, q)$ for all provider peers P:

$$sel(P, q) = \sum_{t \in q} \beta \cdot \log\left(df_{P,k}\right) + (1 - \beta) \cdot \log\left(tf_{P,k}^{max}\right) \tag{7.1}$$

The parameter β in Equation 7.1 can be chosen in the range between 0 and 1. It is used to stress the importance of df or tf^{max}. Preliminary experiments have shown that $\beta = 0.5$ is an appropriate value in most cases [BMT+05a]. Using the scoring function presented above, providers can be can identified that store high-quality documents for query q, and thus achieve high recall by querying only a few providers in the MinervaDL network. To further improve recall, there are peer selection strategies that consider overlap-awareness [BMT+05a] and key correlations [MBN+06].

7.4.2 Behavior Prediction

The prediction mechanism collects IR statistics from the distributed directory and treats them as *time series* data to perform statistical analysis over them. A statistical analysis assumes that the data points taken over time have some sort of internal structure (e.g., trend etc.), and uses this observation to analyze older values and predict future ones [Cha04]. There exist various approaches (see 2.4) that differ in their assumptions about the internal structure of the time-series (e.g., whether it exhibits a *trend* or *seasonality*). *Moving average techniques* as described in 2.4.4.1 are a well-known group of time series prediction techniques that assign equal weights to past observations (e.g., averaging is the simplest form of moving average techniques), and thus cannot cope with trends or seasonality. In the Minerva DL setting, it is reasonable to put more emphasis on a peer's recent behavior and thus assign higher weights to recent observations. *Double exponential smoothing* assigns exponentially decreasing weights to past observations and assumes that the time series data present some trend to predict future values. Since many queries are expected to be short-lived so that no seasonality will be observed in the IR statistics time series, seasonality is not considered in the predictions (and thus, *triple exponential smoothing* is not used).

The scoring function $pred(P, cq)$ returns for a score representing the probability that provider P will publish in the future documents relevant to the continuous query cq. MinervaDL uses *double exponential smoothing* to predict the following two statistical values. Initially, for all keys k in cq, the approach predicts the value for $df_{P,k}$ (denoted as $\hat{df}_{P,k}$), and uses the difference (denoted as $\delta(\hat{df}_{P,k})$) between $\hat{df}_{P,k}$ and the last received value from the directory to calculate the score for P. $\delta(\hat{df}_{P,k})$ reflects the number of relevant documents concerning t that P will publish in the next period. Secondly, MinervaDL predicts $\delta(\hat{cs})$ as the difference in the collection size of P reflecting the provider peer's overall expected future publishing activity. In this way two aspects of the peer's behavior are modeled: its ability to publish relevant documents in the future and its overall expected publishing activity. The consumer peer that submits cq obtains the IR statistics that are needed as an input to the prediction mechanism by utilizing the distributed directory. The following formula is used to compute the prediction score for a provider P with respect to a continuous query cq:

$$pred(P, cq) = \sum_{t \in cq} \log \left(\delta(\hat{df}_{P,k}) + \log\left(\delta(\hat{cs}_P) + 1\right) + 1 \right) \tag{7.2}$$

In the above formula, publication of relevant documents is accented compared to publishing rates. If a peer P publishes no documents at all, or, to be exact, the prediction of $\delta(\hat{cs}_P)$, and thus the prediction of $\delta(\hat{df}_{P,k})$ for all $k \in cq$, is 0 then the $pred(P, cq)$ value is also 0. The addition of 1 in the log formulas is used to yield positive predictions and to avoid $\log 0$.

7.5 Experimental Evaluation

This section presents the evaluation of *MinervaDL*. In the experimental evaluation, a total number of 1000 providers and the same number of consumers is assumed. All these peers are connected to a MinervaDL network using 400 super-peers as access points. At bootstrapping, each provider stores about 300 documents and is mainly specialized in one out of ten categories: *Music*, *Finance*, *Arts*, *Sports*, *Natural Science*, *Health*, *Movies*, *Travel*, *Politics*, and *Nature*. Thus, there are 100 specialized information sources per category.

Overall, the experimental runs use 30 queries[2] containing two, three or four keys that are strong representatives of a certain document category and are used both as one-time and continuous queries. The next sections investigate search performance, filtering performance, and a message cost analysis.

7.5.1 Search Performance

In Figure 7.3, the experimental results of the information retrieval functionality of MinervaDL are presented. Consumers issue the 30 one-time queries and the evaluation measures the *relative recall* to a centralized search engine hosting the complete document collection: the ratio of top-25 documents for a query included in the merged P2P query result. Thus, MinervaDL applies the resource selection strategy as described before in 7.4.1 and increase the percentage ρ of providers in the system that are requested to answer the query.

Figure 7.3 shows that the retrieval performance of MinervaDL manages to retrieve more than 90% of the top rated documents by asking only a fraction of the providers, whereas the baseline approach of *random peer selection* reaches a much lower recall (around 20%). This experiment shows that a random selection of provider peers does not lead to satisfying retrieval results.

7.5.2 Filtering Performance

To evaluate the filtering functionality of MinervaDL, the experiments assume that providers publish 30 documents within a specific time period (called publishing *round*) and the results after ten publishing rounds are investigated. A scenario that models periods of inactivity in the publishing behavior is considered. This scenario assumes that consumers subscribe with 30 continuous queries and reposition them in each publishing round. In the experiments, the percentage ρ of monitored providers is varied and the providers are ranked using the two scoring functions approaches described in Section 7.4. *Recall* is used as filtering measures; it is defined as the ratio of total number of notifications received by the consumers to the total number of published documents matching a subscription.

[2]Example queries are *museum modern art*, or *space model*.

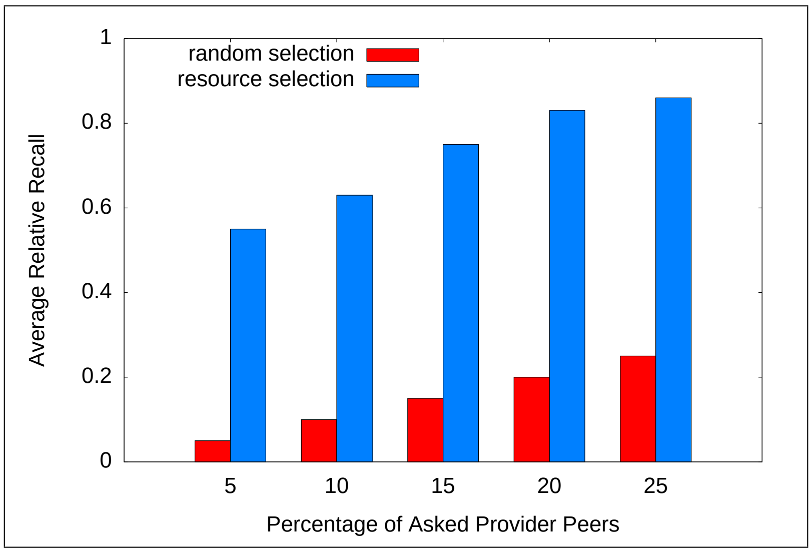


Figure 7.3: Search Performance of MinervaDL.

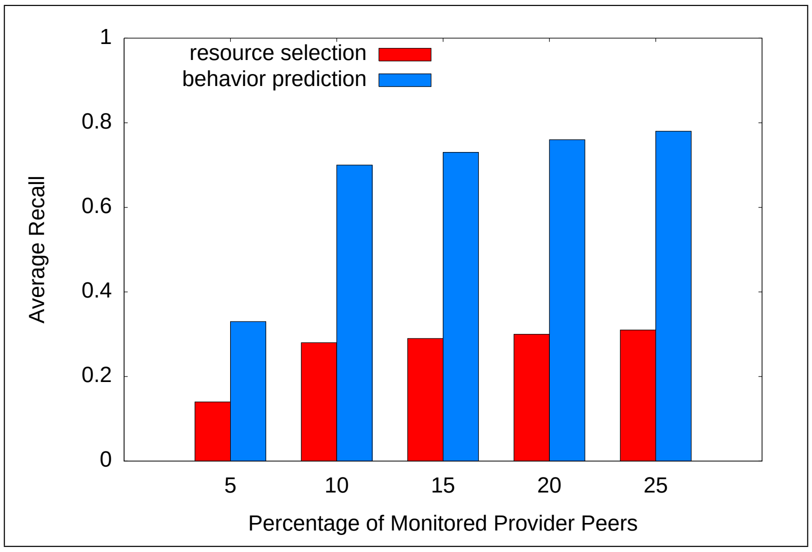


Figure 7.4: Filtering Performance of MinervaDL.

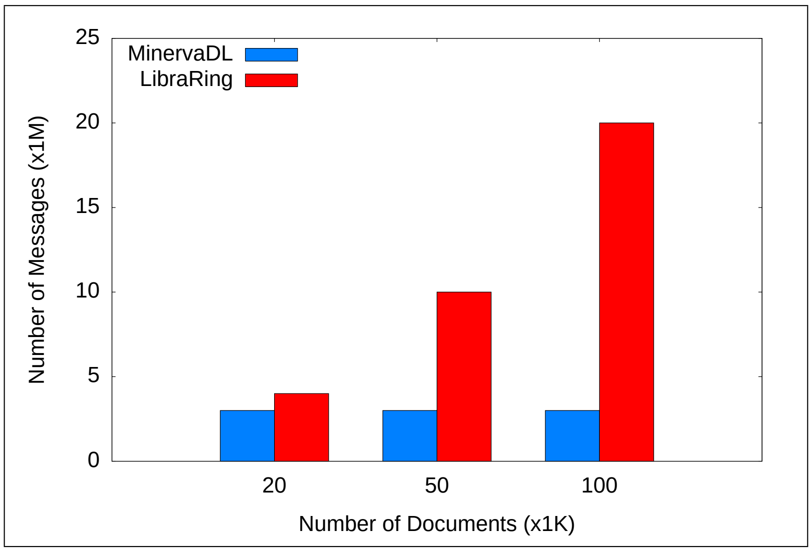


Figure 7.5: Approximate vs. Exact Information Filtering.

As illustrated in Figure 7.4, the use of *behavior prediction* improves recall over *resource selection* as it manages to model more accurately the publishing behavior of providers. In this way, the proposed scoring function manages to achieve a recall of around 80% by monitoring only 20% of the providers in the MinervaDL network. Notice that for resource selection, this number is only 35% which makes it an inapplicable solution to a filtering environment.

7.5.3 Message Costs Analysis

Figures 7.5 and 7.6 show different aspects of a message costs analysis of *MinervaDL*. Here, in each round, the subscriptions are repositioned and the one-time queries are requested again (e.g., by another consumer peer). In this setting, there are three types of messages: *directory*, *retrieval*, and *filtering messages*. Figure 7.5 compares MinervaDL with an implementation of the exact matching protocols of LibraRing [TIK05a].

The overall message costs are considered as a function of the total number of documents published in the system for both approaches. As shown, message costs of MinervaDL are independent of the number of publications, whereas LibraRing, as with all exact matching approaches, is *sensitive* to this parameter since documents have to be disseminated to the network at publication time. This imposes a higher network cost especially for high publication rates as those expected in a DL setting.

Finally, in Figure 7.6, it can be observed that directory messages dominate both retrieval and filtering messages. This is expected since matching and filtering mechanisms are by design local to accommodate high publication rates. This means that building both IR and IF functionality on top of the routing infrastructure imposes no extra cost on the architecture, compared to one that supports only one type of functionality. Since other types of information can also be stored in the directory in the same fashion (e.g., QoS statistics or load balancing information) building extra functionality will increase the added value of the directory maintenance, and come at almost no extra cost.

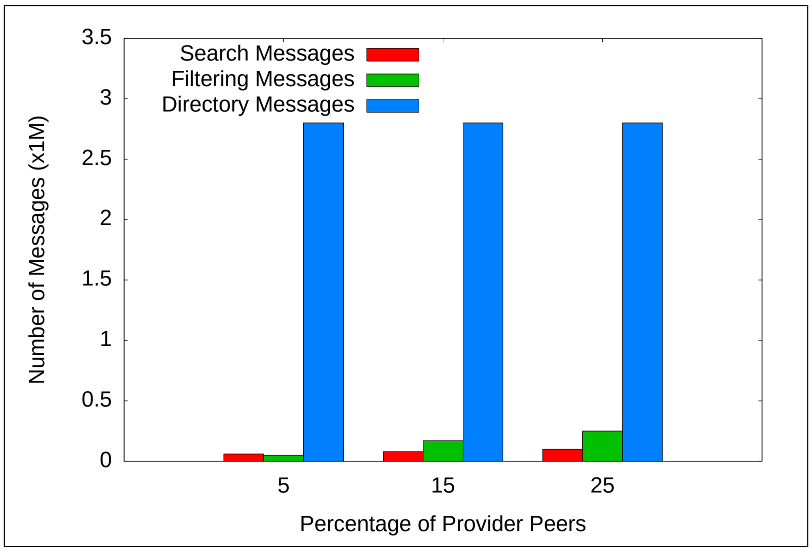


Figure 7.6: Message Costs in MinervaDL.

7.6 Discussion

This chapter presented a digital library use case utilizing a novel DL architecture coined MinervaDL, designed to provide both IR and IF functionality in a single unifying framework. All introduced protocols regulate the communication between the three peer types (super-peers, providers, and consumers). The experimental evaluation of this use case investigated the influence of the individual scoring functions used to perform the two functionalities of MinervaDL. Furthermore, MinervaDL was compared against the LibraRing architecture designed to provide exact retrieval and filtering. The next chapter will conclude this thesis by summarizing the main contributions, and giving an outlook of open problems.

Chapter 8

Conclusion and Open Questions

The last chapter of this doctoral thesis summarizes the presented work (Section 8.1), and lists the main contributions (Section 8.2). Finally, Section 8.3 discusses possible extensions to the presented work and elaborates on open research questions with regard to load balancing, dynamics in P2P networks, replication & caching, and economic aspects[1].

8.1 Summary

Recently, the P2P paradigm has moved towards *distributed data management systems* that are able to offer *information retrieval* or *information filtering* functionality. The main advantage of P2P is its ability to handle huge amounts of data in a decentralized and self-organizing manner. The characteristics of P2P offer high potential benefit for information systems powerful regarding scalability, efficiency, and resilience to failures and dynamics. Beyond that, such an information management system can potentially benefit from the intellectual input of a large user community participating in the data sharing network. Finally, P2P information systems can also bring forward pluralism in informing users about Internet content, a crucial property that breaks information-resource monopolies and bypasses the biased visibility of content from economically powerful sources. In the above distributed and dynamic setting, *information filtering*, also referred to as *publish/subscribe* or *continuous querying* or *information push*, can be seen as equally important to *information retrieval* (or *one-time querying*), since users are able to subscribe to information sources and be notified when new events of interest happen. A user posts a *subscription* (or *continuous query*) to the system to receive *notifications* whenever matching events of interest take place. Information filtering and information retrieval are often referred as *two sides of the same coin* [BC92].

The main challenge addressed in this thesis was to exploit P2P technology for efficient and approximate information filtering. While there exist several approaches to perform exact information filtering in P2P environments, the work in this thesis emphasized a novel architecture to support content-based *approximate* information filtering. Most information filtering approaches taken so far have the underlying hypothesis of potentially delivering notifications from *all* information producers. This exact pub/sub functionality has proven expensive for such distributed environments. The approximate approach in this thesis relaxed this assumption by monitoring only selected sources that are likely to publish documents relevant to the user interests in the future. Since in an IF scenario the data is originally highly distributed residing on millions of sites (e.g., with people contributing to *blogs*), a P2P approach seems an ideal candidate for such a setting.

[1]This chapter focuses on the most important open research questions.

The system presented here offers the ability to identify the top-k most appropriate publisher peers for a given continuous query, i.e., those publishers that are expected to provide the best documents matching it in the future. In this thesis, this task was referred to as *publisher peer selection*. To select this set of peers, this thesis introduced new strategies based on already known *resource selection* techniques in combination with new *behavior prediction* techniques. These new techniques utilize a distributed directory to predict future publishing behavior of peers by applying prediction techniques on *time series* of IR statistics.

8.2 Contributions

The present thesis gives the following four main contributions that have been discussed in detail in the previous chapters:

8.2.1 Approximate Information Filtering

One of the main contributions of this thesis was to introduce the novel MAPS architecture to support content-based approximate information filtering in P2P environments. Most IF approaches so far have the underlying hypothesis of potentially delivering notifications from *every* information producer. Contrary, MAPS relaxes this assumption and monitors only selected sources that are likely to publish documents relevant to the user interests in the future. In MAPS, a user subscribes with a continuous query and only published documents from these monitored sources are forwarded to him. The system provides a network-agnostic P2P architecture with different services and its related protocols (directory, subscription, publication, and notification protocol) for supporting *approximate* IF functionality in a distributed P2P environment. It is the first approach that looks into the problem of approximate IF in such a setting by exploiting metadata stored in a conceptually global, but physically distributed directory. The most critical task in approximate IF is the selection of appropriate publisher peers to meet the information demand in the future. Therefore, MAPS combines existing *resource selection* techniques with new *behavior prediction* strategies. The thesis showed that existing resource selection approaches (also referred to as collection or database selection) are not sufficient in a dynamic filtering setting since resource selection can only determine appropriate authorities that have already published matching documents in the past.

A novel method aiming at behavior prediction of publisher peers complements the peer selection strategy by applying prediction techniques to *time series* of IR statistics. So, MAPS introduces research of time series analysis to P2P information filtering environments. The experimental evaluation of MAPS approved the effectiveness and efficiency in several settings using real Web data. Various publishing behaviors have been investigated to conclude that only the combination resource selection and behavior prediction allows to improve recall, while monitoring only a small number of publishers.

Finally, the aforementioned prediction method was further enhanced by an automatic parameter tuning component. The MAPS Selective Method introduces an automatic method to tune prediction parameters by utilizing known values to adjust the parameters used in the time-series analysis. This is achieved at no additional communication cost, and is a local task performed at each peer individually. To demonstrate the potential gaining from the usage of MAPS, the thesis compared it against an existing exact information filtering approach in the P2P setting and studied several major characteristics (e.g., load balancing issues, query placement, or routing infrastructure).

8.2.2 Correlation Awareness

The second contribution of this thesis covers publisher peer selection for a given continuous query with multiple keywords. For scalability reasons, MAPS's summaries are stored in the distributed directory and have publisher (rather than document) granularity, thus capturing the best publisher for certain *keys*. This, together with per-key organization of the directory that disregards keyword correlations (also referred to as correlated *key sets*) are two of the basic reasons that may lead to low recall. However, considering statistics for *all* possible key sets is clearly not possible due to the explosion in the feature space. In the baseline approach, a continuous query is decomposed into individual keys, and the statistics from the directory are used to compute a combined score for each publisher. This score would represent the probability of each source to publish documents matching the information demand in the near future. This approach may lead to poor filtering quality as the top-ranked publishers for the *complete* query may not be among the top selected publishers. In the worst case, a selected publisher may deliver many documents for each single keyword, but no single document matching *all keywords*, since this information is not present in the directory.

To overcome his problem, this thesis introduced two approaches that use correlations among keys to improve filtering quality: (i) the *USS* (*Unique Synopses Storage*) algorithm that uses existing single-key synopses stored in the directory to estimate the publishing behavior of information sources for key sets, and (ii) the *CSS* (*Combined Synopses Storage*) that enhances the directory to explicitly maintain statistical metadata about selectively chosen key sets. Contrary to distributed IR settings for one-time searching where sources are ranked according to their document collections (i.e., using *resource selection* strategies), in approximate IF the publishers are ranked according to their probability to publish relevant documents in the near future, which poses different requirements for maintaining metadata. This thesis presented the first work to develop algorithms for exploiting keyword correlations in such a dynamic IF setting. Existing and self-limited approaches for two-key queries have been extended to the case of multi-key continuous queries for an arbitrary number of keys. Beyond that, new algorithms to approximate multi-key statistics by combining the statistics of arbitrary subsets have been provided. Hash sketches have been used in similar algorithms with an IR emphasis to compactly represent the documents. This choice has however yielded inaccurate results when considering continuous queries with more than two keys. For this reason the usage of very recent state-of-the-art techniques (KMV synopses) for compact representation of multisets is proposed and applied to an IF setting. These new structures allow the system to compute accurate synopses for multi-key queries, and improve the filtering effectiveness. The experimental evaluation of both algorithms illustrated the filtering performance improvements in comparison to the basic MAPS approach. All experimental series used two different real-world collections for Web and blog data, and applied real-world queries from *Google Zeitgeist*. The evaluation also investigated filtering performance gains depending on the introduced correlation measure (*conditional probability*) representing a way to compute the relatedness among keys.

8.2.3 Prototype System

In addition to the architecture and algorithms for approximate information filtering presented above, the thesis has also developed a prototype implementation of the approximate information filtering approach of MAPS. This implementation was meant to serve as a testing environment for conducted experiments, but also serve as a prototype that is able to demonstrate the applicability of the proposed techniques.

The approximate information filtering approach introduced in this thesis has been integrated into the Minerva search prototype [BMT+05b] such that Minerva provides an approximate publish/subscribe functionality in addition. In this regard, the implementation aspects concerning the extension of Minerva have been investigated and new components to support IF in addition to IR have been implemented and integrated into the existing prototype. An extensive showcase has been used to describe the application of the extended Minerva prototype by executing sample one-time and continuous queries in detail.

8.2.4 Digital Library Use Case

Finally, the thesis presented a digital library use case using the *MinervaDL* architecture as an application scenario to support *approximate* information retrieval and filtering functionality in a single unifying framework. MinervaDL is hierarchical and utilizes a *distributed hash table* to achieve scalability, fault-tolerance, and robustness in its routing layer.

The MinervaDL architecture allows handling huge amounts of data provided by DLs in a distributed and self-organizing way, and provides an infrastructure for creating large networks of digital libraries with minimal administration costs. There are two kinds of basic functionality that are offered in the DL architecture of MinervaDL: (i) an information retrieval scenario (or *one-time querying*, where a user poses an *one-time query* and the system returns all resources matching the query (e.g., all currently available documents relevant to the requested query), and (ii) an information filtering scenario (or *publish/subscribe* or *continuous querying*), where a user submits a *continuous query* (or *subscription*) and waits to be notified from the system about certain events of interest that take place (i.e., about newly published documents relevant to the continuous query).

The proposed DL architecture is built upon a distributed directory storing metadata. The thesis presents routing protocols for information filtering and retrieval that use this directory information to perform the two functionalities. MinervaDL introduces three main components. *Super-peers* run the DHT protocol and form a distributed directory maintaining metadata about providers' local knowledge in compact form. *Consumers* are utilized by users to connect to the MinervaDL network, using a single super-peer as their access point. *Providers* are implemented by information sources that want to expose their content to the MinervaDL network. Typical examples of provider peers are digital libraries of large institutions, like research centers or content providers.

Finally, this thesis presented different scoring functions to select appropriate provider peers answering a one-time query or storing a continuous query for future matching publications. The experimental evaluation investigated the influence of the individual scoring functions used to perform the two functionalities of MinervaDL. Furthermore, MinervaDL was compared to another DL architecture for P2P networks (*LibraRing*) providing retrieval and filtering at the same time. Unlike MinervaDL, this architecture provides exact retrieval and filtering functionality.

8.3 Open Questions

In this section, a list of interesting open problems for approximate information filtering is presented. The topics discussed are relevant not only within the scope of this thesis, but are also interesting in the context of P2P data management in general. The discussion will focus on topics relevant to the efficiency and effectiveness of data management, and will not go into security or privacy issues that, although of high importance, are independent of the proposed solutions.

Thus, the remainder of this section will discuss open questions in the fields of *load balancing*, *dynamics*, *economics*, and *replication & caching*.

8.3.1 Load Balancing

The first open problem related to the thesis is *load balancing* in a distributed P2P setting. Load balancing can be defined as distributing processing and communication activity evenly across a computer network so that no single device is overwhelmed. In a P2P system, the data load of a peer is usually determined by the amount of stored data items per peer. The total data load of the P2P system is defined as the sum of the loads of all peers participating in the system. To talk about a load balanced network, the load of a peer should be around $1/n$ of the total load when considering a network of n peers. A peer is referred to as *overloaded* or *heavy* if it has a significantly higher load compared to the load it would have in a load balanced case. Similarly, a peer is *light* if it has significantly less load than the load in the load balanced case.

As presented in the previous chapters of this thesis, the MAPS approach utilizes distributed hash tables (e.g., Chord [SMK+01] or Pastry [RD01a]) as underlying infrastructure thus exploiting existing work on load balancing techniques for this kind of overlay network. The literature distinguishes two particular problems related to load balancing in distributed hash tables:

- *Address-space load balancing* tries to partition the address-space of the DHT *evenly* among keys. This problem can be seen as solved by relying on *consistent hashing* (i.e., main property of DHT overlay networks) and constructions such as *virtual peers*. With *virtual peers*, each real peer pretends to be several distinct peers, each participating independently in the DHT protocol. The load of a peer is thus determined by summing the load of all its virtual peers. The Chord DHT [SMK+01] is based on *consistent hashing* and needs $O(\log n)$ copies to be operated for every peer [RLS+03]. Drawbacks of virtual peers are increased storage requirement and higher network bandwidth demand. In [KR06], the authors solve the drawbacks by activating only one of the virtual peers at any given time. The work presented in [BKadH05] considers the task of keeping a system balanced in presence of peers leaving and joining the network. The goals is to keep the lengths of intervals assigned to peers differ at most by a constant factor. The proposed scheme works in a constant number of rounds and achieves an optimal balance with high probability.

- *Item load balancing* aims to directly balance the distribution of items among peers in the network. Here, an arbitrary load distribution of items is assumed, i.e., some keys are more popular or appear more frequent than others. [ADH05] and [KR06] are prominent work addressing this issue. A simple and cost-effective approach for item load balancing is presented in [BCM03] applying the *power of two choices* paradigm. There, an item is stored at the less loaded of two or more random alternatives. [RPW04] presents a simple approach for balancing stored data items between peers in a fashion analogous to the dissipation of heat energy in materials. This algorithm improves the distribution of load in DHTs without requiring major changes of the DHTs themselves. In addition, the fault tolerance of P2P systems is increased by the proposed algorithm. Another approach for load balancing is described in [SVF08] and considers differences in peer capacities. In addition, load balance in heavily loaded P2P systems is investigated.

Regarding the MAPS approach presented in this thesis, load balancing affects *address-space* and *item load balancing* at the level of the distributed metadata directory. Beyond that, the distribution of active subscriptions is also important since highly monitored publishers may become a bottleneck. Overall, the following load balancing questions arise and have to be addressed in future work:

- The distributed directory stores metadata concerning publishers for keys or key sets (see Chapter 5). Some keys (or key sets) are more frequent than others resulting in an unbalanced directory distribution. A combination of address-space and item (or key) load balancing techniques will be able to reach a good distribution among keys on directory peers. Nevertheless, metadata lists for certain frequent keys are much longer than lists for infrequent keys since publishers disseminate more directory messages for these keys. Thus, assuming an evenly distributed address-space and an evenly distributed key space is unrealistic, since some directory peers have to store more metadata than others. This issue calls for load balancing techniques that take the frequency of keys into consideration. A possible solutions to load balanced directory entries is to store metadata for a small number of keys (perhaps only for one single key) that are very frequent in one directory peer, whereas another directory peer is assigned to metadata for a high number of very infrequent key.

- Maintaining metadata for key sets in the directory containing more than one key adds additional complexity to the problem of key load balancing. But, key sets can be reduced to single keys since a deterministic mapping from key sets to one single key is considered in Chapter 5. Thus, solving the problem of load balancing for key sets depends on providing solutions for load balancing for single keys.

- Besides the key distribution in the published data sets or documents, the key distribution in the requested continuous queries also influences the directory load distribution. There is no direct relation between these two key distributions since users may request both frequent and infrequent keys.

- Clearly, information filtering in P2P networks is a dynamic process, and publication rates and query rates are not constant but rather change over time. Thus, the distribution of keys in published metadata and the distribution of keys in continuous queries is dynamic too. This calls for load balancing techniques that take changes of key distributions over time into consideration.

- Load balancing is also important regarding active subscriptions. Publisher peers that store many subscriptions for requested continuous queries have to send a high number of notifications for new published documents that match these subscriptions. Thus, load balancing depends also on the number of publications since each publisher peer can adjust the number of publications to its capacities. Decreasing the publication rate also decreases the load for sending notifications because publishers with a low publication rate are less monitored.

Load balancing in P2P information management systems, including the presented MAPS approach, is a very important research topic for future work. Although existing balancing techniques can be applied, there still exist some additional issues that need more sophisticated balancing strategies.

8.3.2 Dynamics

Another open problem for future work involves the effect of *dynamics* in P2P networks. Within a P2P network, the *churn rate*[2] refers to the number of peers leaving the system during a given time period (e.g., an hour, or a year). Churn is a significant problem for large-scale systems, as they must maintain consistent information about peers in the system in order to operate effectively. For example, with a large churn rate it may be impossible to index file locations, making some files inaccessible even though the peers that store these files are online. In addition to peers leaving the network, dynamics in such a setting also includes peers joining or even changing content in an abrupt way.

[KEAAH05] presents an analytical study of churn using a master-equation-based approach using the Chord DHT. For any churn and stabilization rate, and any system size, the authors predict the fraction of failed or incorrect successor and finger pointers (see Section 2.1.3). Earlier work [RGRK04, CCR04, LNBK02] considered theoretical studies of asymptotic performance bounds of DHTs under churn or simulation-based studies.

Discussing the influence of dynamics for the MAPS architecture has to consider several aspects including the directory maintenance, the subscriber and publisher behavior. Since publishing documents is a main part of the information filtering setting, data dynamics is not a concern in MAPS. Thus dynamics in MAPS only covers leaving and joining peers:

- The impact of publishers that leave or join the MAPS network is small. Whenever a publisher joins the network, it has to update the metadata stored in the distributed directory. If it is not the first time that this publisher joins (e.g., the digital library had some downtime), the publishing behavior is recognizable. Subscriptions from previous participation are still valid for future publications. Publishers leaving the MAPS network can not publish documents any more until the next join. A subscriber is not able to send a query to a publisher that has left the network. So, the number of monitored publishers needs to be increased (depending on the expected or estimated number of inactive publishers) to ensure a predefined number of monitored publishers.

- Similarly to the previous discussion, the impact of leaving or joining subscribers is minor. Whenever a subscriber leaves the network, it can not request new continuous queries, although Its active subscriptions will stay valid and issue notifications for matching publications. Of course, the subscriber will not receive the notification, so protocols that support storing notifications in the directory seem to be a necessary system component (see Section 7.3). Then, a subscriber rejoining the MAPS network receives the stored notifications for matching publications during its downtime.

- The effect of peer churn on directory maintenance is a challenging problem that might affect both the effectiveness and efficiency of the system. If a new directory peer joins the MAPS system, it is responsible for a certain subset of the key space. Thus, there are two possibilities to handle joins: (i) the directory peer *waits* for updated metadata from publishers, but until then, the directory cannot provide metadata for certain keys; (ii) the joining protocol includes the dissemination of existing metadata by exploiting the *consistent hashing* property. In this context, a critical issue that arises is the departure of directory peers from the network without prior notification or without following the network disconnection protocol. If so, the metadata for keys that the directory peer was responsible for are no longer available. In this case, publisher peers have to update their statistics to ensure a proper directory maintenance.

[2]The phrase is based on the English idiom *to churn up*, meaning to agitate or produce violent motion

A possible solution for the problem of churn on directory maintenance is replication: more than one directory peers may store metadata for a key to ensure the availability of directory statistics. This issue will be discussed in the next section.

8.3.3 Replication & Caching

An interesting open question for future work involves dealing with *replication & caching*.

Replication is the process of sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices, or computation replication if the same computing task is executed many times. Typically, the access to a replicated entity is uniform with access to a single, non-replicated entity. The replication itself should be transparent to an external user. Also, in a failure scenario, a failover of replicas is hidden as much as possible.

In computer science, a *cache* is a collection of data duplicating original values stored elsewhere or computed earlier, where the original data is expensive to fetch (owing to longer access time) or to compute, compared to the cost of reading the cache. In other words, a cache is a temporary storage area where frequently accessed data can be stored for rapid access. Once the data is stored in the cache, future use can be made by accessing the cached copy rather than re-fetching or recomputing the original data, so that the average access time is shorter. Cache, therefore, helps expedite data access that the CPU would otherwise need to fetch from main memory.

Both caching and replication are also two popular topics in distributed information management. [RL05] compares two popular redundancy schemes: *replication* and *erasure encoding*. Replication is also considered in both DHT protocols used in MAPS (*Chord* [RFH+01] and *Pastry* [RD01b]) whereas erasure encoding is investigated in [DLS+04, KBC+00]. In general, replication denotes the simple redundancy scheme where k identical copies of each data object are kept in the system. The erasure encoding redundancy scheme means that each data object divided into m fragments is recorded into n fragments stored separately with $n > m$. The key property of erasure encoding is that the original data object can be reconstructed from any m fragments. In [ZBW08], a cache-aware, multi-round query routing strategy is developed that balances between query efficiency and result quality in distributed P2P information retrieval (e.g., for the *Minerva* architecture). In addition to this *Exact Caching* (EC) strategy, the aggressive reuse of the cached results of even subsets of a query is proposed as the *Approximate Caching* (AC) technique. This strategy can drastically reduce the bandwidth overheads. Older work proposes a *Uniform Index Caching* (UIC) approach where query results are cached in all peers along the inverse query path such that the same query requested by other peers can be replied from their nearby cached results. The DiCAS (*Distributed Caching and Adaptive Search*) approach [WXLZ04] extends this by distributing the cached results among neighboring peers and queries are forwarded to peers with high probability of providing the desired cache results. Both UIC and DiCAS protocols cache the final results of a query and reuse them only if a query matches exactly the cached results, i.e., only if the results were cached due to an identical query issued earlier. It should be noted that UIC and DiCAS are developed in the context of unstructured P2P networks and are not directly applicable in DHT-based structured networks. In [BCG+03], the authors present a way to efficiently maintain a distributed version of inverted list intersections that were computed frequently to answer queries. Here, the cacheable entity is a materialized intersection of inverted lists which reduces the bandwidth consumption due to frequent shipping of inverted lists. Similar ideas are also proposed recently in [SA06].

In the context of P2P information filtering, it is not obvious either how or what to cache or replicate. In the setting of the thesis, the most benefits out of replication and caching emerge from applying them to the distributed directory of statistics to ensure directory maintenance for high dynamics. In a combined architecture as *MinervaDL*, caching strategies as mentioned above should be applied. Other caching and replication strategies need more work in the future.

8.3.4 Economics

Finally, an interesting direction for future research covers the economic aspects of the presented approximate information filtering approach. In contrast to existing exact filtering approaches, publishers have to disseminate their data items or documents such that the data is available in the system. In MAPS, publishers are not required to expose their actual content to the rest of the network, but rather they disseminate metadata about it. This new paradigm allows to introduce a *payment model* for approximate filtering which introduces a new research direction and an interesting business model.

The design of an incentive-based *payment model* has to consider several cost aspects and restrictions, e.g., the MAPS system has to avoid that subscribers increase the number of monitored publishers above a certain threshold. This can be achieved by introducing a cost for submitting a subscription to a publisher. In the following, the properties of a possible payment model are driven by the following assumptions: (i) subscribers are interested in harvesting information while minimizing their costs, and (ii) publishers are interested in selling information in order to maximize their profit.

An interesting dimension introduced by adding an economic perspective involves the limitation of subscribers to monitor only a few publishers. This limitation makes publisher selection a crucial issue, since subscribers may afford to monitor only a few sources. In the following, interesting issues introduced by an economic model are elaborated:

- Directory peers have to maintain metadata of publishers and provide this information to subscribers that submit a continuous query. One possible payment model could assume that publishers have to pay for storing metadata and subscribers have to pay for receiving metadata. Thus, directory peers have an incentive to store as much metadata as possible, and they are only restricted by their storage capacity and computational power. This assumption ensures that peers not interested in taking part in the directory maintenance get paid.

- Subscribers are interested in receiving documents and data items matching their information demand. Thus, for each continuous query, subscribers have to collect metadata from the directory to perform publisher selection. To avoid harvesting metadata too frequently, subscribers may be charged for that service. In the second step of the payment model, subscribers send the continuous query to providers to store the information demand. Under this assumption, subscribers have to pay for storing their continuous queries to prevent them from subscribing to a high number of publishers. If a subscriber does not receive enough or receives unsatisfying notifications from a publisher, the payment ends and the continuous query is removed.

- Publishers may also need to pay to disseminate their metadata to the directory, since this improves the probability to get selected for storing a continuous query. If a publisher stores a continuous query, and publishes a new document matching this request, a notification is send to the subscribers. Under this model, the publisher may charge on a pay-per-view basis, and the subscriber might choose or not to view a relevant publication. The notification in this case is used by the subscriber to check the meta information concerning a published document or data item and does not include the full publication but only an excerpt or abstract in the form of search engine result presentation.

This simple payment model needs to consider some more issues, e.g., computational power restrictions. Currently, there are efforts to introduce economic interactions into the MAPS architecture.

Appendix A

Abbreviations

Here, the appendix presents a list of abbreviations used in the present doctoral thesis. All abbreviations of table A.1 have been introduced in the previous chapters.

P2P	peer-to-peer
IF	information filtering
IR	information retrieval
DHT	distributed hash table
DL	digital library
MAPS	Minerva Approximate Publish/Subscribe
MSM	MAPS Selective Method
MRS	MAPS Resource Selection
SES	single exponential smoothing
DES	double exponential smoothing
TES	triple exponential smoothing
USS	Unique Synopses Storage
CSS	Combined Synopses Storage
TTL	time-to-live
TA	threshold algorithm
KMV	k minimum values
AKMV	augmented k minimum values
CMS	content management system
df	document frequency
tf	term frequency
HDK	highly discriminative keys
DTF	decision theoretic framework
UIC	Uniform Index Caching
EC	Exact Caching
AC	Approximate Caching
GUI	Graphical User Interface
VSM	vector space model
LSI	latent semantic indexing

Table A.1: List of Abbreviations used in the Thesis.

Appendix B

Zeitgeist Queries

This chapter in the appendix lists the set of used *Zeitgeist* queries. the list of queries from [NBMW06] is used, and distinguishes one-, two-, and three-key queries as shown in Tables B.1, B.2, and B.3. Notice that all queries are stemmed using *Porter Stemmer*, and *stop words* are removed.

3	argo	arsen	bbc	beyonc	bit	car
cat	cbbc	cbeebi	dictionari	dog	dot	dolphin
easter	easyjet	eminem	expedia	fish	florida	flower
footbal	franc	game	gorilla	ibiza	jennif	jordan
kenya	lingeri	britain	live8	liverpool	london	madagascar
mcfli	multimap	o2	oasi	orang	paintbal	playboy
pope	ryanair	simpson	spain	mobil	tesco	killer
tsunami	u2	usher	vodafon	wimbledon	eastend	

Table B.1: 55 One-Key Queries from *Zeitgeist* Query-Log.

50 cent	abi titmuss	alton tower	angelina joli
bbc news	bbc sport	big brother	blink 182
brad pitt	brian mcfadden	british airway	britney spear
charlott church	crazi frog	currenc convert	david beckham
ebay uk	face parti	friend reunit	girl aloud
good charlott	grand nation	green day	harri potter
inland revenu	jennif ellison	jessica alba	johnni depp
london marathon	love heart	manchest unit	michael jackson
paris hilton	radcliff paula	robbi william	star war
wed crasher	train time	valentin day	valentin gift
war world			

Table B.2: 41 Two-Key Queries from *Zeitgeist* Query-Log.

celebr love island	charli chocol factori	red nose day

Table B.3: 3 Three-Key Queries from *Zeitgeist* Query-Log.

Appendix C

Acknowledgements

Writing a doctoral thesis needs assistance and supervision. Thus, sincere thanks are given to all that helped me to make this doctoral thesis during the last years.

- First of all, I have to thank *Prof. Dr.-Ing. Gerhard Weikum* for his support throughout the years of my PhD. His experiences and cooperation improved many ideas. In the last years, I enjoyed the great team work with *Dr. Christos Tryfonopoulos.* It was a great pleasure to work together in such a friendly atmosphere. I have to say thank you to all members of the doctoral committee, Prof. Dr. Jens Dittrich, Prof. Dr. Manolis Koubarakis, and Dr.-Ing. Ralf Schenkel, for their comments and suggestions on improvements and extensions of this work.
- Last years, I was allowed to work in a great research group. So, many thanks to all members of the *Databases and Information Systems Department* for the nice and international atmosphere. A special thank goes to the *secretary team* and the whole *Max-Planck Institute for Informatics* for providing a professional research infrastructure.
- Special thanks go to *my family* and *my parents* for their love and encouragement throughout all my life endeavors. Their courage in life and their strength in difficulties was an extra motivation for this effort.

Bibliography

[Abe01] Karl Aberer. P-Grid: A Self-Organizing Access Structure for P2P Information Systems. In Carlo Batini, Fausto Giunchiglia, Paolo Giorgini, and Massimo Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2001. 20, 28, 85

[Ac04] Yanif Ahmad and Ugur Çetintemel. Networked Query Processing for Distributed Stream-Based Applications. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 456–467. Morgan Kaufmann, 2004. 39

[ADH05] Karl Aberer, Anwitaman Datta, and Manfred Hauswirth. Multifaceted Simultaneous Load Balancing in DHT-Based P2P Systems: A New Game with Old Balls and Bins. In Özalp Babaoglu, Márk Jelasity, Alberto Montresor, Christof Fetzer, Stefano Leonardi, Aad P. A. van Moorsel, and Maarten van Steen, editors, *Self-star Properties in Complex Information Systems, Conceptual and Practical Foundations*, volume 3460 of *Lecture Notes in Computer Science*, pages 373–391. Springer, 2005. 141

[AF00] Mehmet Altinel and Michael J. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 53–64. Morgan Kaufmann, 2000. 28

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993. 89

[AT05] Ioannis Aekaterinidis and Peter Triantafillou. Internet Scale String Attribute Publish/Subscribe Data Networks. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 44–51. ACM, 2005. 29, 85, 119

[AT06] Ioannis Aekaterinidis and Peter Triantafillou. PastryStrings: A Comprehensive Content-Based Publish/Subscribe DHT Network. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), 4-7 July 2006, Lisboa, Portugal*, page 23. IEEE Computer Society, 2006. 29, 38, 39

[BAS04] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In Raj Yavatkar, Ellen W. Zegura, and Jennifer Rexford, editors, *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA*, pages 353–366. ACM, 2004. 29

[BC92] Nicholas J. Belkin and W. Bruce Croft. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM (CACM)*, 35(12):29–38, 1992. 9, 27, 37, 137

[BCFM00] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-Wise Independent Permutations. *Journal of Computer and System Sciences (JCSS)*, 60(3):630–659, 2000. 53

[BCG$^+$03] Bobby Bhattacharjee, Sudarshan S. Chawathe, Vijay Gopalakrishnan, Peter J. Keleher, and Bujor D. Silaghi. Efficient Peer-To-Peer Searches Using Result-Caching. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers*, volume 2735 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 2003. 144

[BCM03] John W. Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple Load Balancing for Distributed Hash Tables. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers*, volume 2735 of *Lecture Notes in Computer Science*, pages 80–87. Springer, 2003. 141

[BHR$^+$07] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. On Synopses for Distinct-Value Estimation Under Multiset Operations. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 199–210. ACM, 2007. 35, 86

[BK07] Nilesh Bansal and Nick Koudas. Searching the Blogosphere. In *Tenth International Workshop on the Web and Databases, WebDB 2007, Beijing, China, June 15, 2007*, 2007. 93

[BKadH05] Marcin Bienkowski, Miroslaw Korzeniowski, and Friedhelm Meyer auf der Heide. Dynamic Load Balancing in Distributed Hash Tables. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV, 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005, Revised Selected Papers*, volume 3640 of *Lecture Notes in Computer Science*, pages 217–225. Springer, 2005. 141

[BKK+03] Hari Balakrishnan, M. Frans Kaashoek, David R. Karger, Robert Morris, and Ion Stoica. Looking Up Data in P2P systems. *Communications of the ACM (CACM)*, 46(2):43–48, 2003. 16

[Blo70] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM (CACM)*, 13(7):422–426, 1970. 53

[BM96] Timothy A. H. Bell and Alistair Moffat. The Design of a High Performance Information Filtering System. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 12–20. ACM, 1996. 28

[BMPC07] Matthias Bender, Sebastian Michel, Josiane Xavier Parreira, and Tom Crecelius. P2P Web Search: Make It Light, Make It Fly (Demo). In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 164–168. www.crdrdb.org, 2007. 105

[BMS+06] Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Martin Theobald, and Gerhard Weikum. IO-Top-k: Index-access Optimized Top-k Query Processing. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 475–486. ACM, 2006. 27

[BMT+05a] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Improving Collection Selection with Overlap Awareness in P2P Search Engines. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 67–74. ACM, 2005. 38, 41, 42, 46, 53, 106, 108, 122, 131

[BMT+05b] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. MINERVA: Collaborative P2P Search. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 1263–1266. ACM, 2005. 12, 106, 107, 122, 140

[BMTW06] Matthias Bender, Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Global Document Frequency Estimation in Peer-to-Peer Web Search. In *Ninth International Workshop on the Web and Databases, WebDB 2006, Chicago, Illinois, USA, June 30, 2006*, 2006. 108

[BMW06] Matthias Bender, Sebastian Michel, and Gerhard Weikum. P2P Directories for Distributed Web Search: From Each According to His Ability, to Each According to His Needs. In Roger S. Barga and Xiaofang Zhou, editors, *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, page 51. IEEE Computer Society, 2006. 71

[BMWZ04] Matthias Bender, Sebastian Michel, Gerhard Weikum, and Christian Zimmer. Bookmark-driven Query Routing in Peer-to-Peer Web Search. In Jamie Callan, Norbert Fuhr, and Wolfgang Nejdl, editors, *Proceedings of the SIGIR Workshop on Peer-to-Peer Information Retrieval (P2P-IR), 27th Annual International ACM SIGIR Conference, July 29, 2004, Sheffield, UK*, 2004. 108

[BMWZ05] Matthias Bender, Sebastian Michel, Gerhard Weikum, and Christian Zimmer. The MINERVA Project: Database Selection in the Context of P2P Search. In Gottfried Vossen, Frank Leymann, Peter C. Lockemann, and Wolffried Stucky, editors, *Datenbanksysteme in Business, Technologie und Web, 11. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Karlsruhe, 2.-4. März 2005*, volume 65 of *LNI*, pages 125–144. GI, 2005. 38, 53, 54, 106, 108

[BY99] Ricardo Baeza-Yates. *Modern Information Retrieval.* Addison Wesley, 1999. 25

[BYJK$^+$02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting Distinct Elements in a Data Stream. In José D. P. Rolim and Salil P. Vadhan, editors, *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, volume 2483 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2002. 35

[Cal96] James P. Callan. Document Filtering With Inference Networks. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 262–269. ACM, 1996. 28

[Cal98] James P. Callan. Learning While Filtering DFocuments. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 224–231. ACM, 1998. 28

[CAPMN03] Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003), 22-24 June 2003, Seattle, WA, USA*, pages 236–249. IEEE Computer Society, 2003. 105, 118

[CCC$^+$01] Alexis Campailla, Sagar Chaki, Edmund M. Clarke, Somesh Jha, and Helmut Veith. Efficient Filtering in Publish-Subscribe Systems Using Binary Decision. In *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, 12-19 May 2001, Toronto, Ontario, Canada*, pages 443–452. IEEE Computer Society, 2001. 28

[CCH92] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY Retrieval System. In *Database and Expert Systems Applications (DEXA), Proceedings of the International Conference in Valencia, Spain, 1992*, pages 78–83. Springer, 1992. 53

[CCMN00] Moses Charikar, Surajit Chaudhuri, Rajeev Motwani, and Vivek R. Narasayya. Towards Estimation Error Guarantees for Distinct Values. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, May 15-17, 2000, Dallas, Texas, USA*, pages 268–279. ACM, 2000. 33

[CCR04] Miguel Castro, Manuel Costa, and Antony I. T. Rowstron. Performance and Dependability of Structured Peer-to-Peer Overlays. In *2004 International Conference on Dependable Systems and Networks (DSN 2004), 28 June - 1 July 2004, Florence, Italy, Proceedings*, pages 9–18. IEEE Computer Society, 2004. 143

[CDTW00] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In Weidong Chen and Jeffrey F. Naughton and Philip A. Bernstein, editor, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 379–390. ACM, 2000. 39

[CF03] Sirish Chandrasekaran and Michael J. Franklin. PSoup: A System for Streaming Queries Over Streaming Data. *VLDB Journals*, 12(2):140–156, 2003. 39

[CFGR02] Chee Yong Chan, Pascal Felber, Minos N. Garofalakis, and Rajeev Rastogi. Efficient Filtering of XML Documents with XPath Expressions. In *Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA*, pages 235–244. IEEE Computer Society, 2002. 28

[CG05] Graham Cormode and Minos N. Garofalakis. Sketching Streams Through the Net: Distributed Approximate Query Tracking. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 373–384. ACM, 2005. 36

[CGR05] Gianna M. Del Corso, Antonio Gulli, and Francesco Romani. Ranking a Stream of News. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 97–106. ACM, 2005. 28

[Cha02] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002. 24, 25

[Cha04] Christopher Chatfield. *The Analysis of Time Series: An Introduction*. CRC Press, 2004. 29, 33, 54, 131

[CIKN04] Paul-Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl. Publish/subscribe for rdf-based p2p networks. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *Lecture Notes in Computer Science*, pages 182–197. Springer, 2004. 124

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR'95, Proceedings of the 18th Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval. Seattle, Washington, USA, July 9-13, 1995, pages 21–28. ACM Press, 1995. 53, 54, 108, 129, 131

[CMH+02] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing (IC)*, 6(1):40–49, 2002. 19

[CRW01] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Database Systems (TODS)*, 19(3):332–383, 2001. 28

[CW04] Pei Cao and Zhe Wang. Efficient Top-K Query Calculation in Distributed Networks. In Soma Chaudhuri and Shay Kutten, editors, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 206–215. ACM, 2004. 106

[DAF+03] Yanlei Diao, Mehmet Altinel, Michael J. Franklin, Hao Zhang, and Peter M. Fischer. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. *ACM Transactions on Database Systems (TODS)*, 28(4):467–516, 2003. 28

[Den82] Peter J. Denning. Electronic Junk. *Communications of the ACM (CACM)*, 25(3):163–165, 1982. 27

[DF03] Marianne Durand and Philippe Flajolet. Loglog Counting of Large Cardinalities (Extended Abstract). In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, volume 2832 of *Lecture Notes in Computer Science*, pages 605–617. Springer, 2003. 33, 34

[DFK05] Jens-Peter Dittrich, Peter M. Fischer, and Donald Kossmann. AGILE: Adaptive Indexing for Context-Aware Information Filters. In Fatma Özcan, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 215–226. ACM, 2005. 119

[DLS+04] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. Designing a DHT for Low Latency and High Throughput. In *1st Symposium on Networked Systems Design and Implementation (NSDI 2004), March 29-31, 2004, San Francisco, California, USA, Proceedings*, pages 85–98. USENIX, 2004. 144

[EH05] Wolfgang Müller 0002, Martin Eisenhardt, and Andreas Henrich. Scalable Summary Based Retrieval in P2P Networks. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 586–593. ACM, 2005. 118

[Fag02] Ronald Fagin. Combining Fuzzy Information: an Overview. *ACM SIGMOD Record*, 31(2):109–118, 2002. 118

[FFS$^+$01] Daniel Faensen, Lukas Faulstich, Heinz Schweppe, Annika Hinze, and Alexander Steidinger. Hermes: A Notification Service for Digital Libraries. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2001, Roanoke, Virginia, USA, June 24-28, 2001, Proceedings*, pages 373–380. ACM, 2001. 41, 119

[FJL$^+$01] Françoise Fabret, Hans-Arno Jacobsen, François Llirbat, João Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe. In *ACM SIGMOD Conference 2001: Santa Barbara, CA, USA*, pages 115–126. ACM, 2001. 28

[FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal Aggregation Algorithms for Middleware. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*. ACM, 2001. 26

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic Counting Algorithms for Data Base Applications. *Journal of Computer and System Sciences (JCSS)*, 31(2):182–209, 1985. 33

[FPC$^+$99] James C. French, Allison L. Powell, James P. Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the Performance of Database Selection Algorithms. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 238–245. ACM, 1999. 129

[FSGM$^+$98] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. Computing Iceberg Queries Efficiently. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 299–310. Morgan Kaufmann, 1998. 89

[Fuh99] Norbert Fuhr. A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Transactions on Information Systems (TOIS)*, 17(3):229–249, 1999. 25, 53

[FZ98] Michael J. Franklin and Stanley B. Zdonik. Data In Your Face: Push Technology in Perspective. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 516–519. ACM Press, 1998. 28

[GBK00] Ulrich Güntzer, Wolf-Tilo Balke, and Werner Kießling. Optimizing Multi-Feature Queries for Image Databases. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 419–428. Morgan Kaufmann, 2000. 26

[GDH04] Evgeniy Gabrilovich, Susan T. Dumais, and Eric Horvitz. Newsjunkie: Providing Personalized Newsfeeds via Analysis of Information Novelty. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 482–490. ACM, 2004. 28

[GGG$^+$03] P. Krishna Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The Impact of DHT Routing Geometry on Resilience and Proximity. In *Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 25-29, 2003, Karlsruhe, Germany*, pages 381–394, 2003. 20

[GGM95] Luis Gravano and Hector Garcia-Molina. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pages 78–89. Morgan Kaufmann, 1995. 53, 108

[GGMT99] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. GlOSS: Text-Source Discovery over the Internet. *ACM Transactions on Database Systems (TODS)*, 24(2):229–264, 1999. 53, 129, 131

[Gib01] Phillip B. Gibbons. Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports. In Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 541–550. Morgan Kaufmann, 2001. 33

[GL03] Bugra Gedik and Ling Liu. PeerCQ: A Decentralized and Self-Configuring Peer-to-Peer Information Monitoring System. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003), 19-22 May 2003, Providence, RI, USA*, pages 490–499. IEEE Computer Society, 2003. 29, 39

[GSAA04] Abhishek Gupta, Ozgur D. Sahin, Divyakant Agrawal, and Amr El Abbadi. Meghdoot: Content-Based Publish/Subscribe over P2P Networks. In Hans-Arno Jacobsen, editor, *Middleware 2004, ACM/IFIP/USENIX International Middleware Conference, Toronto, Canada, October 18-20, 2004, Proceedings*, volume 3231 of *Lecture Notes in Computer Science*, pages 254–273. Springer, 2004. 29, 39

[GT02] Daniel I. Greenstein and Suzanne E. Thorin. *The Digital Library: A Biography*. Digital Library Federation and Council on Library and Information Resources, 2002. 123

[GWJD03] Leonidas Galanis, Yuan Wang, Shawn R. Jeffery, and David J. DeWitt. Locating Data Sources in Large Distributed Systems. In Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer, editors, *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 874–885. Morgan Kaufmann, 2003. 117

[Ham94] James Douglas Hamilton. *Time Series Analysis*. Princeton University Press, 1994. 29

[HPS96] David A. Hull, Jan O. Pedersen, and Hinrich Schütze. Method Combination For Document Filtering. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 279–287. ACM, 1996. 27

[HT99] David Hawking and Paul B. Thistlewaite. Methods for Information Server Selection. *ACM Transactions on Information Systems (TOIS)*, 17(1):40–76, 1999. 105

[HWBM02] Cefn Hoile, Fang Wang, Erwin Bonsma, and Paul Marrow. Core Specification and Experiments in DIET: A Decentralised Ecosystem-inspired Mobile Agent System. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*, pages 623–630. ACM, 2002. 119

[IKT04a] Stratos Idreos, Manolis Koubarakis, and Christos Tryfonopoulos. P2P-DIET: An Extensible P2P Service that Unifies Ad-hoc and Continuous Querying in Super-Peer Networks. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 933–934. ACM, 2004. 28, 119, 124

[IKT04b] Stratos Idreos, Manolis Koubarakis, and Christos Tryfonopoulos. P2P-DIET: One-Time and Continuous Queries in Super-Peer Networks. In *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004, Proceedings*, volume 2992 of *Lecture Notes in Computer Science*, pages 851–853. Springer, 2004. 28

[JHR$^+$04] Ankur Jain, Joseph M. Hellerstein, Sylvia Ratnasamy, , and David Wetherall. A Wakeup Call for Internet Monitoring Systems: The Case for Distributed Triggers. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *Third Workshop on Hot Topics in Networks (HotNets-III), November 15-16, 2004, San Diego, CA USA*, 2004. 39

[KBC$^+$00] John Kubiatowicz, David Bindel, Yan Chen, Steven E. Czerwinski, Patrick R. Eaton, Dennis Geels, Ramakrishna Gummadi, Sean C. Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Y. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *ASPLOS-IX Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, November 12-15, 2000.*, pages 190–201, 2000. 144

[KEAAH05] Supriya Krishnamurthy, Sameh El-Ansary, Erik Aurell, and Seif Haridi. A Statistical Theory of Chord Under Churn. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV, 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005, Revised Selected Papers*, volume 3640 of *Lecture Notes in Computer Science*, pages 93–103. Springer, 2005. 143

[KKTR02] Manolis Koubarakis, Theodoros Koutris, Christos Tryfonopoulos, and Paraskevi Raftopoulou. Information Alert in Distributed Digital Libraries: The Models, Languages, and Architecture of DIAS. In Maristella Agosti and Costantino Thanos, editors, *Research and Advances Technology for Digital Technology : 6th European Conference, ECDL 2002, Rome, Italy, September 16-18, 2002. Proceedings*, volume 2458 of *Lecture Notes in Computer Science*, pages 527–542. Springer, 2002. 28, 119

[Kle00] Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA. ACM, 2000*, pages 163–170, 2000. 19

[KLFW06] Nils Kammenhuber, Julia Luxenburger, Anja Feldmann, and Gerhard Weikum. Web Search Clickstreams. In Jussara M. Almeida, Virgílio A. F. Almeida, and Paul Barford, editors, *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement 2006, Rio de Janeriro, Brazil, October 25-27, 2006*, pages 245–250. ACM, 2006. 108

[KLL$^+$97] David R. Karger, Eric Lehman, Frank Thomson Leighton, Rina Panigrahy, Matthew S. Levine, and Daniel Lewin. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 654–663. ACM, 1997. 29

[KP05] Georgia Koloniari and Evaggelia Pitoura. Peer-to-Peer Management of XML Data: Issues and Research Challenges. *ACM SIGMOD Record*, 34(2):6–17, 2005. 28

[KR06] David R. Karger and Matthias Ruhl. Simple Efficient Load-Balancing Algorithms for Peer-to-Peer Systems. *Theory of Computing Systems*, 39(6):787–804, 2006. 141

[KTID03] Manolis Koubarakis, Christos Tryfonopoulos, Stratos Idreos, and Yannis Drougas. Selective Information Dissemination in P2P Networks: Problems and Solutions. *ACM SIGMOD Record*, 32(3):71–76, 2003. 28

[LC03] Jie Lu and James P. Callan. Content-based Retrieval in Hybrid Peer-to-Peer Nnetworks. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pages 199–206. ACM, 2003. 121

[LCC00] Leah S. Larkey, Margaret E. Connell, and James P. Callan. Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 6-11, 2000*, pages 282–289. ACM, 2000. 45

[LdS01] Carl Lagoze and Herbert Van de Sompel. The Open Archives Initiative: Building a Low-barrier Interoperability Framework. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2001, Roanoke, Virginia, USA, June 24-28, 2001, Proceedings*, pages 54–62. ACM, 2001. 123

[LKP+05] Toan Luu, Fabius Klemm, Ivana Podnar, Martin Rajman, and Karl Aberer. Alvis Peers: A Scalable Full-Text Peer-to-Peer Retrieval Engine. In *Proceedings of the International Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR 2006), Arlington, Virginia, USA, November 11, 2006*, pages 41–48. ACM, 2005. 105, 118

[LLH+03] Jinyang Li, Boon Thau Loo, Joseph M. Hellerstein, M. Frans Kaashoek, David R. Karger, and Robert Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers*, volume 2735 of *Lecture Notes in Computer Science*, pages 207–215. Springer, 2003. 106

[LNBK02] David Liben-Nowell, Hari Balakrishnan, and David R. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing (PODC), July 21-24, 2002 Monterey, California, USA. ACM, 2002*, pages 233–242, 2002. 22, 143

[LPT00] Ling Liu, Calton Pu, and Wei Tang. Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12(5):861, 2000. 39

[Luh58] H. P. Luhn. A Business Intelligence System. *IBM Journal of Reasearch and Development*, 2(4):314–319, 1958. 27

[MBN+06] Sebastian Michel, Matthias Bender, Nikos Ntarmos, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P Routing Indices. In Philip S. Yu, Vassilis J. Tsotras, Edward A. Fox, and Bing Liu, editors, *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 172–181. ACM, 2006. 38, 46, 53, 86, 87, 88, 103, 107, 108, 131

[MBTW06] Sebastian Michel, Matthias Bender, Peter Triantafillou, and Gerhard Weikum. IQN Routing: Integrating Quality and Novelty in P2P Querying and Ranking. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, volume 3896 of *Lecture Notes in Computer Science*, pages 149–166. Springer, 2006. 108

[MLS99] David R. H. Miller, Tim Leek, and Richard M. Schwartz. A Hidden Markov Model Information Retrieval System. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 214–221. ACM, 1999. 53

[MMK+05] Volker Markl, Nimrod Megiddo, Marcel Kutsch, Tam Minh Tran, Peter J. Haas, and Utkarsh Srivastava. Consistently Estimating the Selectivity of Conjuncts of Predicates. In *Proceedings of the 31st International Conference* *on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 373–384. ACM, 2005. 92

[MNR02] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *PODC 2002, Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, July 21-24, 2002 Monterey, California, USA. ACM, 2002*, pages 183–192, 2002. 20

[MS94] Masahiro Morita and Yoichi Shinoda. Information Filtering Based on User Behaviour Analysis and Best Match Text Retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin,Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 272–281. ACM/Springer, 1994. 27

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. 25

[MS05] Peter Mahlmann and Christian Schindelhauer. Peer-to-Peer Networks Based on Random Transformations of Connected Regular Undirected Graphs. In *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms, July 18-20, 2005, Las Vegas, Nevada, USA*, pages 155–164, 2005. 20

[MSHR02] Samuel Madden, Mehul A. Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously Adaptive Continuous Queries Over Streams. In Michael J. Franklin, Bongki Moon, and Anastassia Ailamaki, editors, *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pages 49–60. ACM, 2002. 39

[MTW05] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. KLEE: A Framework for Distributed Top-k Query Algorithms. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 637–648. ACM, 2005. 106

[MYL02] Weiyi Meng, Clement T. Yu, and King-Lup Liu. Building Efficient and Effective Metasearch Engines. *ACM Computing Surveys (CSUR)*, 34(1):48–89, 2002. 24, 53

[NACP01] Benjamin Nguyen, Serge Abiteboul, Gregory Cobena, and Mihai Preda. Monitoring XML Data on the Web. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 437–448. ACM, 2001. 28

[NBMW06] Thomas Neumann, Matthias Bender, Sebastian Michel, and Gerhard Weikum. A Reproducible Benchmark for P2P Retrieval. In Philippe Bonnet and Ioana Manolescu, editors, *Proceedings of the First International Workshop on Performance and Evaluation of Data Management Systems, ExpDB 2006, in cooperation with ACM SIGMOD, June 30, 2006, Chicago, Illinois, USA*, pages 1–8, 2006. 93, 149

[NF03] Henrik Nottelmann and Norbert Fuhr. Evaluating Different Methods of Estimating Retrieval Quality for Resource Selection. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 290–297. ACM, 2003. 53, 108

[NF06] Henrik Nottelmann and Norbert Fuhr. Comparing Different Architectures for Query Routing in Peer-to-Peer Networks. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, Proceedings*, volume 3936 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2006. 53

[NR99] Surya Nepal and M. V. Ramakrishna. Query Processing Issues in Image (Multimedia) Databases. In *Proceedings of the 15th International Conference on Data Engineering (ICDE), 23-26 March 1999, Sydney, Austrialia*, pages 22–29. IEEE Computer Society, 1999. 26

[NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: A P2P Networking Infrastructure based on RDF. In *Proceedings of the Eleventh International World Wide Web Conference, WWW2002, Honolulu, Hawaii, USA, 7-11 May 2002*, pages 604–615. ACM, 2002. 28, 124

[Ora01] Andy Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001. 16

[PB02] Peter R. Pietzuch and Jean Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *22nd International Conference on Distributed Computing Systems, Workshops (ICDCSW '02) July 2-5, 2002, Vienna, Austria, Proceedings*, pages 611–618. IEEE Computer Society, 2002. 28, 39, 122

[PC98] Jay M. Ponte and W. Bruce Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 275–281. ACM, 1998. 25, 53

[PMBW05] Odysseas Papapetrou, Sebastian Michel, Matthias Bender, and Gerhard Weikum. On the Usage of Global Document Occurrences in Peer-to-Peer Information Systems. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, volume 3760 of *Lecture Notes in Computer Science*, pages 310–328. Springer, 2005. 108

[PRL+07] Ivana Podnar, Martin Rajman, Toan Luu, Fabius Klemm, and Karl Aberer. Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys. In *Proceedings of the 23nd International Conference on Data Engineering, ICDE*

2007, April 15-20, 2007, The Marmara Hotel, Istanbul, Turkey, pages 1096–1105. IEEE, 2007. 87

[PRR97] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *SPAA '97: Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures, June 23-25, 1997, Newport, RI, USA. ACM Press*, pages 311–320, 1997. 20

[RD01a] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Rachid Guerraoui, editor, *Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12-16, 2001, Proceedings*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer, 2001. 20, 23, 28, 109, 119, 141

[RD01b] Antony I. T. Rowstron and Peter Druschel. Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP), October 21-24, 2001, Chateau Lake Louise, Banff, Alberta, Canada*, pages 188–201. ACM, 2001. 23, 144

[RFH$^+$01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, 2001, San Diego, CA, USA*, pages 161–172. ACM, 2001. 20, 28, 39, 41, 144

[RGRK04] Sean C. Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT (Awarded Best Paper!). In *Proceedings of the General Track: 2004 USENIX Annual Technical Conference, June 27 - July 2, 2004, Boston Marriott Copley Place, Boston, MA, USA*, pages 127–140. USENIX, 2004. 143

[RKCD01] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. SCRIBE: The Design of a Large-Scale Event Notification Infrastructure. In Jon Crowcroft and Markus Hofmann, editors, *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*, volume 2233 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2001. 23, 28, 119

[RL05] Rodrigo Rodrigues and Barbara Liskov. High Availability in DHTs: Erasure Coding vs. Replication. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV, 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005, Revised Selected Papers*, volume 3640 of *Lecture Notes in Computer Science*, pages 226–239. Springer, 2005. 144

[RLS$^+$03] Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard M. Karp, and Ion Stoica. Load Balancing in Structured P2P Systems. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22, 2003, Revised Papers*, volume 2735 of *Lecture Notes in Computer Science*, pages 68–79. Springer, 2003. 141

[RPTW08] Paraskevi Raftopoulou, Euripides G.M. Petrakis, Christos Tryfonopoulos, and Gerhard Weikum. Information Retrieval and Filtering over Self-Organising Digital Libraries. In *Research and Advanced Technology for Digital Libraries, 12th European Conference, ECDL 2008, Aarhus, Denmark, September 14-19, 2008, Proceedings*, Lecture Notes in Computer Science. Springer, 2008. 121, 124

[RPW04] Simon Rieche, Leo Petrak, and Klaus Wehrle. A Thermal-Dissipation-Based Approach for Balancing Data Load in Distributed Hash Tables. In *29th Annual IEEE Conference on Local Computer Networks (LCN 2004), 16-18 November 2004, Tampa, FL, USA, Proceedings*, pages 15–23. IEEE Computer Society, 2004. 141

[RV03] Patrick Reynolds and Amin Vahdat. Efficient Peer-to-Peer Keyword Searching. In Markus Endler and Douglas C. Schmidt, editors, *Middleware 2003, ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 16-20, 2003, Proceedings*, volume 2672 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2003. 118

[RvRP80] Stephen E. Robertson, C. J. van Rijsbergen, and Martin F. Porter. Probabilistic Models of Indexing and Searching. In *Proceedings of the 3rd annual ACM conference on Research and Development in Information Retrieval (SIGIR), Cambridge, England*, pages 35–56. ACM, 1980. 25

[RWHB$^+$92] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at trec. In *The First Text REtrieval Conference (TREC). National Institute of Standards and Technology (NIST)*, pages 21–30, 1992. 26

[SA06] Gleb Skobeltsyn and Karl Aberer. Distributed Cache Table: Efficient Query-driven Processing of Multi-term Queries in P2P Networks. In *P2PIR '06: Proceedings of the International Workshop on Information Retrieval in Peer-to-Peer Networks, Arlington, Virginia, USA.*, pages 33–40. ACM, 2006. 144

[SC99] Fei Song and W. Bruce Croft. A General Language Model for Information Retrieval. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, November 2-6, 1999*, pages 316–321. ACM, 1999. 53

[SCL$^+$05] Jeremy Stribling, Isaac G. Councill, Jinyang Li, M. Frans Kaashoek, David R. Karger, Robert Morris, and Scott Shenker. OverCite: A Cooperative Digital Research Library. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV, 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005, Revised Selected Papers*, volume 3640 of *Lecture Notes in Computer Science*, pages 69–79. Springer, 2005. 118

[SE00] Atsushi Sugiura and Oren Etzioni. Query Routing for Web Search Engines: Architecture and Experiments. *Computer Networks*, 33(1-6):417–429, 2000. 105

[SJCO02] Luo Si, Rong Jin, James P. Callan, and Paul Ogilvie. A Language Modeling Framework for Resource Selection and Results Merging. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge*

Management, McLean, VA, USA, November 4-9, 2002, pages 391–397. ACM, 2002. 25, 53, 54, 113

[SLZ$^+$07] Gleb Skobeltsyn, Toan Luu, Ivana Podnar Zarko, Martin Rajman, and Karl Aberer. Web Text Retrieval with a P2P Query-Driven Index. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 679–686. ACM, 2007. 87

[SMK$^+$01] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, 2001, San Diego, CA, USA*, pages 149–160. ACM, 2001. 20, 28, 42, 43, 85, 109, 125, 128, 141

[SMLN$^+$03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003. 122, 130

[SMwW$^+$03] Torsten Suel, Chandan Mathur, Jo wen Wu, Jiangong Zhang, Alex Delis, Mehdi Kharrazi, Xiaohui Long, and Kulesh Shanmugasundaram. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In Vassilis Christophides and Juliana Freire, editors, *International Workshop on Web and Databases, San Diego, California, June 12-13, 2003*, pages 67–72, 2003. 118, 121

[STSW02] Sergej Sizov, Martin Theobald, Stefan Siersdorfer, and Gerhard Weikum. BINGO!: Bookmark-Induced Gathering of Information. In Tok Wang Ling, Umeshwar Dayal, Elisa Bertino, Wee Keong Ng, and Angela Goh, editors, *3rd International Conference on Web Information Systems Engineering (WISE 2002), 12-14 December 2002, Singapore, Proceedings*, pages 323–332. IEEE Computer Society, 2002. 41, 58, 78, 111

[SVF08] Tyler Steele, Vivek Vishnumurthy, and Paul Francis. A Parameter-Free Load Balancing Mechanism For P2P Networks. In *The 7th International Workshop on Peer-to-Peer Systems, IPTPS 2008, Tampa Bay, Florida, USA, February 25-26, 2008*, 2008. 141

[SW05] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*. Springer, 2005. 15, 16

[TAJ04] David Tam, Reza Azimi, and Hans-Arno Jacobsen. Building Content-Based Publish/Subscribe Systems with Distributed Hash Tables. In Karl Aberer, Vana Kalogeraki, and Manolis Koubarakis, editors, *Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003, Revised Papers*, volume 2944 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2004. 28

[TBF$^+$03] Wesley W. Terpstra, Stefan Behnel, Ludger Fiege, Andreas Zeidler, and Alejandro P. Buchmann. A Peer-to-Peer Approach to Content-Based Publish/Subscribe. In Hans-Arno Jacobsen, editor, *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003, Sunday, June 8th, 2003, San Diego, California, USA (in conjunction with SIGMOD/PODS)*. ACM, 2003. 28

[TC91] Howard R. Turtle and W. Bruce Croft. Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems (TOIS)*, 9(3):187–222, 1991. 53

[TD04] Chunqiang Tang and Sandhya Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval. In *1st Symposium on Networked Systems Design and Implementation (NSDI 2004), March 29-31, 2004, San Francisco, California, USA, Proceedings*, pages 211–224. USENIX, 2004. 105, 118

[TE04] Peter Triantafillou and Andreas A. Economides. Subscription Summarization: A New Paradigm for Efficient Publish/Subscribe Systems. In *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, pages 562–571. IEEE Computer Society, 2004. 28

[TGNO92] Douglas B. Terry, David Goldberg, David Nichols, and Brian M. Oki. Continuous Queries over Append-Only Databases. In Michael Stonebraker, editor, *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, June 2-5, 1992*, pages 321–330. ACM Press, 1992. 39

[TIK05a] Christos Tryfonopoulos, Stratos Idreos, and Manolis Koubarakis. LibraRing: An Architecture for Distributed Digital Libraries Based on DHTs. In Andreas Rauber, Stavros Christodoulakis, and A. Min Tjoa, editors, *Research and Advanced Technology for Digital Libraries, 9th European Conference, ECDL 2005, Vienna, Austria, September 18-23, 2005, Proceedings*, volume 3652 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2005. 29, 39, 85, 117, 121, 122, 124, 129, 134

[TIK05b] Christos Tryfonopoulos, Stratos Idreos, and Manolis Koubarakis. Publish/subscribe Functionality in IR Environments using Structured Overlay Networks. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 322–329. ACM, 2005. 29, 38, 39, 40, 47, 58, 70, 71

[TKD04] Christos Tryfonopoulos, Manolis Koubarakis, and Yannis Drougas. Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators. In Mark Sanderson, Kalervo Järvelin, James Allan, and Peter Bruza, editors, *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*, pages 313–320. ACM, 2004. 29, 45, 129

[TKD08] Christos Tryfonopoulos, Manolis Koubarakis, and Yannis Drougas. Information Filtering and Query Indexing for an Information Retrieval Model. *ACM Transactions on Information Systems (TOIS)*, 2008. 29, 45

[TSW05] Martin Theobald, Ralf Schenkel, and Gerhard Weikum. An Efficient and Versatile Query Engine for TopX Search. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 625–636. ACM, 2005. 27

[TX03] Chunqiang Tang and Zhichen Xu. pFilter: Global Information Filtering and Dissemination Using Structured Overlay Networks. In *9th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2003), 28-30 May 2003, San Juan, Puerto Rico, Proceedings*, pages 24–30. IEEE Computer Society, 2003. 29, 38, 39, 40, 85, 119

[TZWK07] Christos Tryfonopoulos, Christian Zimmer, Gerhard Weikum, and Manolis Koubarakis. Architectural Alternatives for Information Filtering in Structured Overlays. *IEEE Internet Computing (IC)*, 11(4):24–34, 2007. 11, 47, 121

[WGD03] Yuan Wang, Leonidas Galanis, and David J. DeWitt. Galanx: An efficient Peer-to-Peer Search Engine System. Technical report, University of Wisconsin, Madison, 2003. 117

[WMYL01] Zonghuan Wu, Weiyi Meng, Clement T. Yu, and Zhuogang Li. Towards a Highly-Scalable and Effective Metasearch Engine. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 386–395, 2001. 105

[WXLZ04] Chen Wang, Li Xiao, Yunhao Liu, and Pei Zheng. Distributed Caching and Adaptive Search in Multilayer P2P Networks. In *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, pages 219–226. IEEE Computer Society, 2004. 144

[XC99] Jinxi Xu and W. Bruce Croft. Cluster-Based Language Models for Distributed Retrieval. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 254–261. ACM, 1999. 53

[YGM94a] Tak W. Yan and Hector Garcia-Molina. Index Structures for Information Filtering Under the Vector Space Model. In *Proceedings of the Tenth International Conference on Data Engineering, February 14-18, 1994, Houston, Texas, USA*, pages 337–347. IEEE Computer Society, 1994. 28

[YGM94b] Tak W. Yan and Hector Garcia-Molina. Index Structures for Selective Dissemination of Information Under the Boolean Model. *ACM Transactions on Database Systems (TODS)*, 19(2):332–364, 1994. 28

[YGM99] Tak W. Yan and Hector Garcia-Molina. The SIFT Information Dissemination System. *ACM Transactions on Database Systems (TODS)*, 24(4):529–565, 1999. 28, 45, 129

[YGM03] Beverly Yang and Hector Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE), March 5-8, 2003, Bangalore, India*, pages 49–, 2003. 20

[YJ06] Beverly Yang and Glen Jeh. Retroactive Answering of Search Queries. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 457–466. ACM, 2006. 37, 122

[YL97] Budi Yuwono and Dik Lun Lee. Server Ranking for Distributed Text Retrieval Systems on the Internet. In Rodney W. Topor and Katsumi Tanaka, editors, *Database Systems for Advanced Applications '97, Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA), Melbourne, Australia, April 1-4, 1997*, volume 6 of *Advanced Database Research and Development Series*, pages 41–50. World Scientific, 1997. 129, 131

[YVGM04] Beverly Yang, Patrick Vinograd, and Hector Garcia-Molina. Evaluating GUESS and Non-Forwarding Peer-to-Peer Search. In *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, pages 209–218. IEEE Computer Society, 2004. 105

[ZBW08] Christian Zimmer, Srikanta Bedathur, and Gerhard Weikum. Flood Little, Cache More: Effective Result-reuse in P2P IR Systems. In Jayant R. Haritsa, Ramamohanarao Katagiri, and Vikram Pudi, editors, *Database Systems for Advanced Applications, 13th International Conference, DASFAA 2008, New Delhi, India, March 2008, Proceedings*, volume 4947 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2008. 108, 144

[ZC01] Yi Zhang and James P. Callan. Maximum Likelihood Estimation for Filtering Thresholds. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 294–302. ACM, 2001. 28

[ZH05] Rongmei Zhang and Y. Charlie Hu. HYPER: A Hybrid Approach to Efficient Content-Based Publish/Subscribe. In *25th International Conference on Distributed Computing Systems (ICDCS 2005), 6-10 June 2005, Columbus, OH, USA*, pages 427–436. IEEE Computer Society, 2005. 39

[ZHTW08] Christian Zimmer, Johannes Heinz, Christos Tryfonopoulos, and Gerhard Weikum. P2P Information Retreival and Filtering with MAPS. In *Eighth IEEE International Conference on Peer-to-Peer Computing (P2P 2008), September 8-12, 2008, Aachen, Germany*. IEEE Computer Society, 2008. 12, 105

[ZL01] ChengXiang Zhai and John D. Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR*

2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA, pages 334–342. ACM, 2001. 53

[ZM06] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Computing Surveys (CSUR)*, 38(2), 2006. 26

[ZTB$^+$07] Christian Zimmer, Christos Tryfonopoulos, Klaus Berberich, Gerhard Weikum, and Manolis Koubarakis. Node Behavior Prediction for Large-Scale Approximate Information Filtering. In *Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR), July 27 2007, Amsterdam, The Netherlands*, 2007. 10, 73

[ZTB$^+$08] Christian Zimmer, Christos Tryfonopoulos, Klaus Berberich, Manoli Koubarakis, and Gerhard Weikum. Approximate Information Filtering in Peer-to-Peer Networks. In James Bailey, David Maier, Klaus-Dieter Schewe, and Bernhard Thalheim, editors, *Web Information Systems Engineering - WISE 2008, 9th International Conference on Web Information Systems Engineering, Auckland, Newzealand, September 1-4, 2008*, volume 5175 of *Lecture Notes in Computer Science*, pages 6–19. Springer, 2008. 10, 51

[ZTW07] Christian Zimmer, Christos Tryfonopoulos, and Gerhard Weikum. MinervaDL: An Architecture for Information Retrieval and Filtering in Distributed Digital Libraries. In László Kovács, Norbert Fuhr, and Carlo Meghini, editors, *Research and Advanced Technology for Digital Libraries, 11th European Conference, ECDL 2007, Budapest, Hungary, September 16-21, 2007, Proceedings*, volume 4675 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2007. 13, 121

[ZTW08] Christian Zimmer, Christos Tryfonopoulos, and Gerhard Weikum. Exploiting Correlated Keywords to Improve Approximate Information Filtering. In *SIGIR 2008: Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Singapore, July 20-24, 2008*. ACM, 2008. 12, 85

[ZZJ$^+$01] Shelley Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination. In *Network and Operating System Support for Digital Audio and Video, 11th International Workshop, NOSSDAV 2001, Port Jefferson, NY, USA, June 25-26, 2001, Proceedings*, pages 11–20. ACM, 2001. 28

Index