Approximate Information Filtering in Peer-to-Peer Networks

Christian Zimmer[1], Christos Tryfonopoulos[1], Klaus Berberich[1], Manolis Koubarakis[2], and Gerhard Weikum[1]
{czimmer, trifon, kberberi, weikum}@mpi-inf.mpg.de, and koubarak@di.uoa.gr

[1]Max-Planck-Institute for Informatics, Saarbrücken, Germany
[2]National and Kapodistrian University of Athens, Greece

Abstract. Most approaches to information filtering taken so far have the underlying hypothesis of potentially delivering notifications from every information producer to subscribers. This exact publish/subscribe model creates an efficiency and scalability bottleneck, and might not even be desirable in certain applications. The work presented here puts forward MAPS, a novel approach to support approximate information filtering in a peer-to-peer environment. In MAPS a user subscribes to and monitors only carefully selected data sources, and receives notifications about interesting events from these sources only. This way scalability is enhanced by trading recall for lower message traffic. We define the protocols of a peer-to-peer architecture especially designed for approximate information filtering, and introduce new node selection strategies based on time series analysis techniques to improve data source selection. Our experimental evaluation shows that MAPS is scalable; it achieves high recall by monitoring only few data sources.[1]

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. In such a dynamic setting, *information filtering (IF)*, also referred to as *publish/subscribe*, *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 Alert or the QSR system [1]. In an IF scenario, a user posts a *subscription* (or *continuous query*) to the system to receive *notifications* whenever certain events of interest take place (e.g., when a paper on distributed systems becomes available).

In this paper we put forward MAPS (*Minerva Approximate Publish/Subscribe*), a novel architecture to support content-based approximate information filtering in peer-to-peer (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

[1] This work has been partially supported by the EU projects AEOLUS and EVERGROW.

that are likely to publish documents relevant to the user's interests in the future. In MAPS, a user subscribes with a continuous query and monitors only the most interesting sources in the network. Only published documents from these sources are forwarded to him. 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), a P2P approach seems an ideal candidate for such a setting. However, exact pub/sub functionality has proven expensive for such distributed environments [2–4]. MAPS offers a natural solution to this problem, by avoiding document granularity dissemination as it is the main scalability bottleneck of 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 an information avalanche and thus cognitive overload to the subscribed users, if the users were alerted for each and every new document published at any source whenever this matched a submitted continuous query. Our approximate IF approach ranks sources, and delivers matches only from the best ones, by utilizing novel publisher selection strategies. Despite that the presented approach focuses on a P2P setting based on *Distributed Hash Tables* (DHT) [21, 19], notice that our 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. In the light of the above, the contributions presented in this paper are threefold:

- We define a network-agnostic P2P architecture and its related protocols for supporting *approximate* IF functionality in a P2P environment. To the best of our knowledge this is the first approach that looks into the problem of approximate IF in such a setting.
- We show that traditional resource selection strategies are not sufficient in this setting, and devise a *novel* method to predict publishing behavior based on time series analysis of IR metrics. This technique allows us to improve recall, while monitoring only a small number of publishers.
- We present an extensive experimental study of our prediction mechanism in terms of recall, and evaluate our protocols in terms of message load in the network.

In previous work, we have compared exact and approximate information filtering in [5], applied approximate IR and IF to the digital library domain [6], and investigated different time series analysis methods [7]. The current paper extends the core ideas behind approximate IF by elaborating on the protocols, discussing in detail our prediction mechanisms, and presenting an extensive experimental evaluation of our approach.

The rest of the paper is organized as follows. Related work is discussed in Section 2. Section 3 presents the MAPS architecture and discusses the related protocols, while Section 4 introduces our node selection method. Experimental results are presented in Section 5, and Section 6 concludes this paper.

2 Related Work

Database research on continuous queries has its origins in [8] and systems like OpenCQ [9] and NiagaraCQ [10]. 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 [11, 12] and distributed proposals [13–15]. The efforts to improve network efficiency and reduce delivery delays in content-based pub/sub systems lead to approaches like HYPER [16], where a hybrid architecture that exploits properties of subject-based pub/sub approaches is presented. Hermes [17] was one of the first proposals to use a Distributed Hash Table (DHT) for building a topic-based pub/sub system, while PeerCQ [13] utilized a DHT to build a content-based system for processing continuous queries. Finally, Meghdoot [18] utilized the CAN DHT [19] to support an attribute-value data model and offered new ideas for the processing of subscriptions with range predicates and load balancing.

Recently, several systems that employed an IR-based query language to support information filtering on top of structured overlay networks have been deployed. DHTrie [3] extended the Chord protocol [21] to achieve exact information filtering functionality and applied document-granularity dissemination to achieve the recall of a centralized system. In the same spirit, LibraRing [20] presented a framework to provide information retrieval and filtering services in two-tier digital library environments. Similarly, pFilter [2] used a hierarchical extension of the CAN DHT [19] to store user queries and relied on multi-cast trees to notify subscribers. In [4], the authors show how to implement a DHT-agnostic solution to support prefix and suffix operations over string attributes in a pub/sub environment.

Query placement, as implemented in exact information filtering approaches such as [2, 3], is deterministic, and depends upon the terms contained in the query and the hash function provided by the DHT. These query placement protocols lead to filtering effectiveness of a centralized system. Compared to a centralized approach, [2, 3] exhibit scalability, fault-tolerance, and load balancing at the expense of high message traffic at publication time. In MAPS, only the most promising nodes store a user query and are thus monitored. Publications are only matched against its local query database, 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.

3 The MAPS Protocols

3.1 The Directory Protocol

MAPS utilizes a conceptually global, but physically distributed directory, which can be layered on top of a Chord-style DHT [21], and manages aggregated information about each node's local knowledge in compact form, similarly to [22]. The DHT partitions the term space, such that every node is responsible for the statistics of a randomized subset of terms within the directory. To keep IR statistics up-to-date, each node distributes per-term summaries of its *local index* along with contact information to the directory. For

efficiency reasons, these messages can be piggy-backed to DHT maintenance messages with applying batching strategies.

To facilitate message sending, we will use the function `send(`msg, I`)` to send message msg to the node responsible for identifier I. This is similar to the Chord function `lookup(`I`)` [21], and costs $O(\log n)$ overlay hops for a network of n nodes. In MAPS, every publisher node uses POST messages to distribute per-term statistics.

Let us now examine how a publisher node P updates the global directory when $T = \{t_1, t_2, \ldots, t_k\}$ denotes the set of all terms contained in document publications of P occurring after the last update. For each term $t_i \in T$, P computes the maximum frequency of occurrence of term t_i within the documents contained in P's collection ($tf_{t_i}^{max}$), the number of documents in the document collection of P that t_i is contained in (df_{t_i}), and the size of the document collection cs. Having collected the statistics for term t_i, P creates message POST$(id(P), ip(P), tf_{t_i}^{max}, df_{t_i}, cs, t_i)$, where $id(P)$ is the identifier of node P and $ip(P)$ is the IP address of P. P then uses function `send()` to forward the message to the node D responsible for identifier $H(t_i)$ (i.e., the node responsible for maintaining statistics for term t_i by using the Chord hash function H mapping terms to identifiers). Once a node D receives a POST message, it stores the statistics for P in its local post database to keep them available on request for any node. Finally, notice that our architecture allows the usage of any type of P2P network (structured or unstructured), given that the necessary information (i.e., the per-node IR statistics) is made available through appropriate protocols.

3.2 The Subscription Protocol

The subscription protocol is responsible for selecting the publishers that will index a query. The node selection procedure utilizes the directory to discover and retrieve node statistics that will guide query indexing. Then, a ranking of the potential sources is performed and the query is sent to the *top-k* ranked publishers. The publishers storing the query are the only ones that will be monitored for new publications.

Let us assume that a subscriber node S wants to subscribe with a *multi-term query* q of the form $t_1 t_2 \ldots t_k$ with k distinct terms. To do so, S needs to determine which nodes 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 node selection strategy described in Section 4.

To collect statistics about the data sources, S needs to contact all directory nodes responsible for the query terms. Thus, for each query term t_i, S computes $H(t_i)$, which is the identifier of the node responsible for storing statistics about other nodes that publish documents containing the term t_i. Subsequently, S creates message COLLECTSTATS$(id(S), ip(S), t_i)$, and uses the function `send()` to forward the message in $O(\log n)$ hops to the node responsible for identifier $H(t_i)$. Notice that the message contains $ip(S)$, so its recipient can directly contact S.

When a node D receives a COLLECTSTATS message asking for the statistics of term t_i, it searches its local post store to retrieve the node list L_i of all posts of the term. Subsequently, a message RETSTATS(L_i, t_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

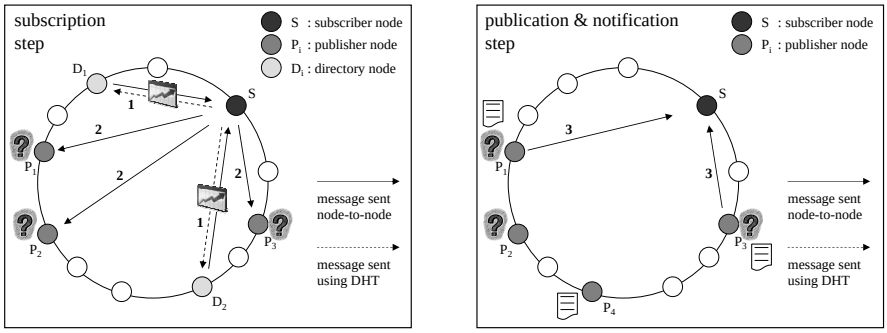

Fig. 1. Illustration of the MAPS protocols.

of Figure 1, where S contacts directory nodes D_1 and D_2. Once S has collected all the node lists L_i for the terms contained in q, it utilizes an appropriate scoring function $score(N, q)$ to compute a node score with respect to q, for each one of the nodes N contained in L_i. Based on the score calculated for each node, a ranking of nodes is determined and the highest ranked nodes are candidates for storing q.

Once the nodes that will store q have been determined, S constructs message INDEXQ$(id(S), ip(S), q)$ and uses the IP addresses associated with the node to forward the message to the nodes 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 [23, 24]. This procedure is shown in step 2 of Figure 1, where S contacts publishers P_1, P_2 and P_3.

Nodes 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 nodes are monitored for publications, the node ranking function becomes a critical component, which will determine the final recall achieved. This scoring function can be based on standard resource selection approaches from the IR literature (e.g. CORI [25]). However, as we show in Section 4.2, these approaches alone are not sufficient in an IF setting, since they where designed for retrieval scenarios, in contrast to the IF scenario considered here, and are aimed at identifying specialized authorities. Filtering and node 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 re-executes the subscription protocol.

3.3 Publication and Notification Protocol

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 NOTIFY$(id(P), ip(P), d)$ and sends it to S using the IP address associated with the stored query (shown in step 3 of Figure 1). Notice that nodes publishing documents relevant to a query q, but not storing it, will produce no notification (e.g., node P_4 in Figure 1).

4 Node Selection Strategy

To select which publishers should be monitored, the subscription protocol of Section 3.2 uses a scoring function to rank nodes. In our approach the subscriber computes a node score based on a combination of *resource selection* and *node behavior prediction* formulas as shown below:

$$score(P, q) = \alpha \cdot sel(P, q) + (1 - \alpha) \cdot pred(P, q) \quad (1)$$

In Equation 1, q is a query, P is a publisher node, and $sel(P, q)$ and $pred(P, q)$ are scoring functions based on resource selection and prediction methods respectively that assign a score to a node P with respect to a query q. Here, $score(P, q)$ is the scoring function that decides the final score. The tunable parameter α affects the balance between authorities (high $sel(P, q)$ scores) and nodes with potential to publish matching documents in the future (high $pred(P, q)$ scores). Based on these scores, a ranking of nodes is determined and q is forwarded to the highest ranked nodes. Notice that our node selection strategy is general and can also be used in centralized settings, where a server (instead of the distributed directory of Section 3.1) maintains the necessary statistics, and mediates the interaction between publishers and subscribers.

To show why an approach that relies only on resource selection is not sufficient, and give the intuition behind node behavior prediction, consider the following example. Assume a node P_1 that has specialized and become an authority in sports, but publishes no relevant documents any more. Another node 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, node P_1 would always be chosen to index the user's query, which would be wrong given that node P_1 no longer publishes sports-related documents. On the other hand, to be assigned a high score by the ranking function, node P_2 would have to specialize in sports – a long procedure that is inapplicable in a IF setting which is by definition dynamic. The fact that resource selection alone is not sufficient is even more evident in the case of news items. News items have a short shelf-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. We employ node behavior prediction to cope with these dynamics. The main contribution of our work with respect to predicting node behavior, is to view the IR statistics as time series and use statistical analysis tools to model node behavior. Time series analysis accounts for the fact that the observations have some sort of internal structure (e.g., trend, seasonality etc.), and uses this fact to analyze older values and predict future ones.

4.1 Time Series Analysis

To predict node behavior, we consider time series of IR statistics, thus making a rich repository of techniques from time series analysis [26] applicable. These techniques

predict future time series values based on past observations and differ in (i) their assumptions about the internal structure (e.g., whether trends and seasonality can be observed) and (ii) their flexibility to put emphasis on more recent observations. Since the considered IR statistics exhibit trends, for instance, when nodes successively crawl sites that belong to different topics, or, gradually change their thematic focus, the employed time series prediction technique must be able to deal with trends. Further, in our scenario we would like to put emphasis on a node's recent behavior and thus assign higher weight to recent observations when making predictions about its future behavior. We choose *double exponential smoothing* (DES) as a prediction technique, since it can both deal with trends and put emphasis on more recent observations. Its explanation is included in detail in [7] in combination with optimized method to apply DES in the MAPS setting.

For completeness we mention that there is also triple exponential smoothing that, in addition, handles seasonality in the observed data. For an application with many long-lasting queries, one could use triple-exponential smoothing, so that seasonality is taken into account.

4.2 Node Behavior Prediction

The function $pred(P, q)$ returns a score for a node P that represents the likelihood of publishing documents relevant to query q in the future. Using the DES technique, two values are predicted. First, for all terms t in query q, we predict the value for $df_{P,t}$ (denoted as $\hat{df}_{P,t}$), and use the difference (denoted as $\delta(\hat{df}_{P,t})$) between the predicted and the last value obtained from the directory to calculate the score for P (function δ signifies difference). Value $\delta(\hat{df}_{P,t})$ reflects the number of relevant documents that P will publish in the next time-unit. Second, we predict $\delta(\hat{cs})$ as the difference in the collection size of node P reflecting the node's overall expected future publishing activity. We thus model two aspects of the node's behavior: (i) its potential to publish relevant documents in the future, and (ii) its overall expected future publishing activity. The time series of IR statistics that are needed as an input to our prediction mechanism are obtained using the distributed directory. The predicted behavior for node P is quantified as follows:

$$pred(P, q) = \sum_{t \in q} \log \Big(\delta(\hat{df}_{P,t}) + \log \left(\delta(\hat{cs}_P) + 1 \right) + 1 \Big) \tag{2}$$

In the above Equation 2, the publishing of relevant documents is more accented than the dampened publishing rate. If a node publishes no documents at all, or, to be exact, $\delta(\hat{cs})$ and $\delta(\hat{df})$ are 0, then the $pred(P, q)$ value is also 0. The addition of 1 in the $\log$ formulas yields positive predictions and avoids $\log 0$.

4.3 Resource Selection

The function $sel(P, q)$ returns a score for a node P and a query q, and is calculated using standard resource selection algorithms from the IR literature (see [27] for an overview), such as tf-idf based methods, CORI [25], or language models. Using $sel(P, q)$, we identify authorities specialized in a topic, which, as argued above, is not sufficient for

our IF setting. In our implementation we use an CORI-like approach well-known from P2P information retrieval [22].

5 Experimental Evaluation

5.1 Experimental Setup

To conduct each experiment described in the next sections the following steps are executed. Initially the network is set up and the underlying DHT is created. Subsequently, subscribers utilize the protocol described in Section 3.2 to subscribe to selected publishers. We will say that a publisher node is *monitored* with query q by a subscriber when it stores q in its local query database. 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 on average per peer. At the end of each round, message costs and recall for this round are calculated, and subscribers rank publishers using Equation 1 to reposition their queries accordingly. We consider the following system parameters:

- ρ**:** The percentage of top-ranked publishers. In experiments, ρ is the same for all subscribers and different system properties for a value of ρ up to 25% are investigated. It is clear that when $\rho = 100\%$ (i.e., all publishers are monitored) then recall is 1, and our approach degenerates to exact filtering.
- α**:** To control the influence of resource selection vs. node behavior prediction in our experiments, we vary the value of α in the node selection formula. A value of α close to 0 emphasizes node behavior prediction, while values close to 1 stress resource selection.

To investigate the effectiveness and efficiency of our approach, we model node publishing behavior through different publishing scenarios described in Section 5.3. Retrieval effectiveness of our approach is utilized by recall, while efficiency is measured using a benefit/cost ratio metric. Both are defined as follows:

- **Recall:** We measure recall by computing the *ratio* of the total number of notifications received by subscribers to the total number of published documents matching subscriptions. In experiments we consider the *average recall* computed over *all* rounds (i.e., for the complete experiment).
- **Benefit/Cost Ratio:** To evaluate the efficiency of our approach, we measure the total number of *subscription* and *notification* messages to calculate the benefit/cost ratio as the number of notifications per message sent. Notice that in our approach no publication messages are needed, since publications trigger only local node computations and are not disseminated as in exact matching approaches.

As explained in Section 3, the number of subscription messages depends on the number of query terms 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 5.3 and 5.4, the message cost needed to maintain the distributed directory information is not taken into account since our main goal is to focus on the filtering protocol costs. Typically, directory messages are included in DHT maintenance messages, thus they can be considered as part of the underlying routing infrastructure.

5.2 Experimental Data

The data collection contains over 2 million documents from a focused Web crawl 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. The number of distinct terms after stemming amounts to $593,876$.

In all experiments, the network consists of $1,000$ nodes containing 300 documents each in their initial local collection. Each peer hosts 15% random documents, 10% not categorized documents, and 75% documents from a single category, resulting in 100 nodes specializing in each category. Using the document collection, we construct 30 continuous queries containing two, three or four query terms. Each of the query terms selected is a strong representative of a document category (i.e., a frequent term 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*.

5.3 Different Publishing Scenarios

To measure MAPS's efficiency in terms of recall and message cost under various settings, we consider four scenarios representing different publishing behaviors. The overall number of published documents is constant in all scenarios ($300K$ documents) therefore the maximum number of notifications concerning the 30 active queries is also constant ($146,319$ notifications), allowing us to compare across different scenarios. The following figures show 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 node selection method is included for comparison purposes.

Consistent Publishing The first publishing scenario targets the performance of our approach when nodes' interests remain unchanged over time. Figure 2 shows that the average recall and the benefit/cost ratio do not depend on the ranking method used, and our 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 node 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 nodes. Compared to the baseline approach of random selection, our approach achieves

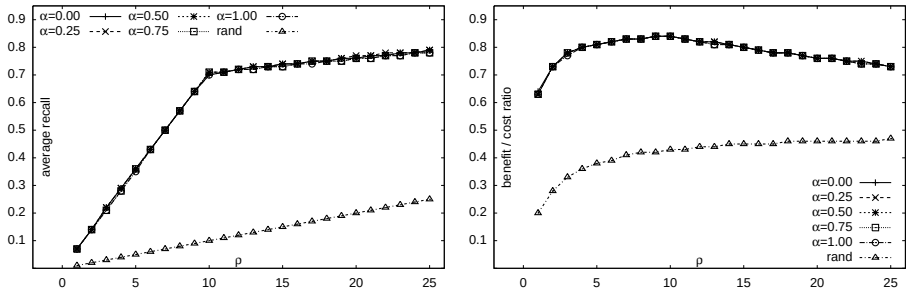

Fig. 2. Average recall and benefit/cost ratio for *Consist* scenario.

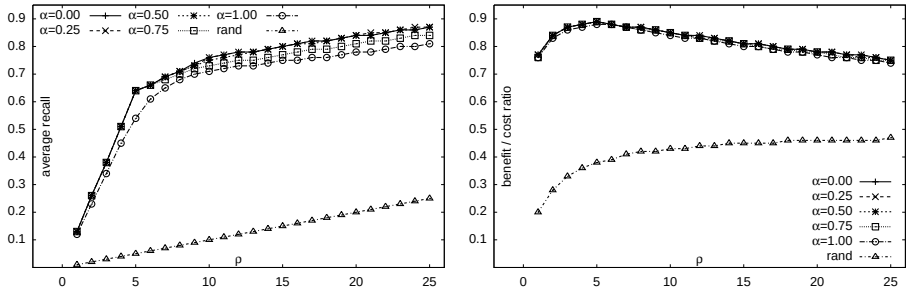

Fig. 3. Average recall and benefit/cost ratio for *CatChg* scenario.

up to 7 times a higher average recall (for $\rho = 10\%$). Finally, the best value for the benefit/cost ratio is when $\rho = 10\%$.

Category Change Since users may publish documents from different topics, we use this scenario to simulate the changes in a publisher's content. In the *CatChg* scenario, a node initially publishes documents from one category, and switches to a different category at some point in time. Figures 3 illustrates the performance of our approach in this scenario for different values for α and ρ. The most important observation from this figure 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). In general, both average recall and benefit cost/ratio improve as α reaches 0 and prediction is stressed. This abrupt changes in the publishers' content cannot be captured by the resource selection method, which favors topic authorities. On the other hand, publication prediction detects the publishers' topic change from changes in the IR statistics and adapts the scoring function to monitor nodes publishing documents relevant to subscribed queries.

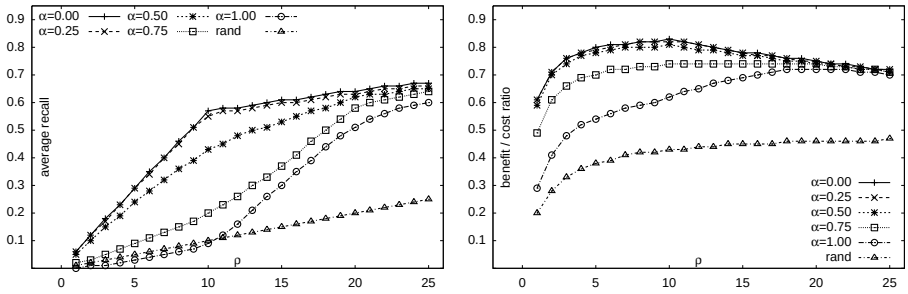

Fig. 4. Average recall and benefit/cost ratio for *Break* scenario.

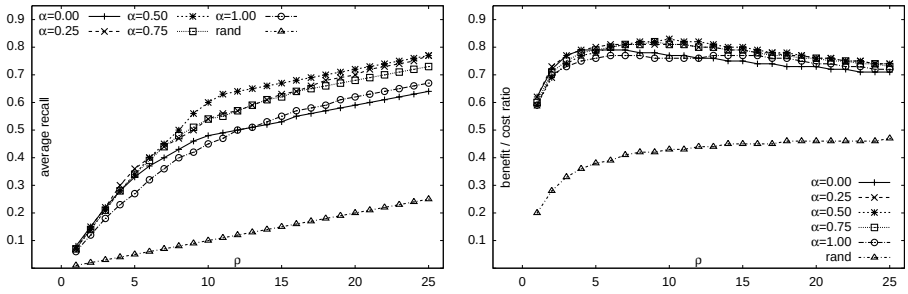

Fig. 5. Average recall and benefit/cost ratio for *TmpChg* scenario.

Publishing Breaks The *Break* scenario models the behavior of nodes as they log in and out of the network. We assume that some publisher is active and publishes documents for some rounds, and then logs 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. Our ranking mechanism should adapt to these inactivity periods, and distinguish between nodes not publishing documents any more and nodes making temporary pauses.

Figure 4 demonstrates that both average recall and benefit/cost ratio improve when resource selection is emphasized (i.e., when α is close to 1), since pauses in the publishing mislead the prediction formula to foresee that, in the future, no relevant publications will occur. For this reason, nodes 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. Consequently, selecting a ranking method that favors prediction leads to poor recall and low benefit/cost ratio, that are comparable to those of *rand*.

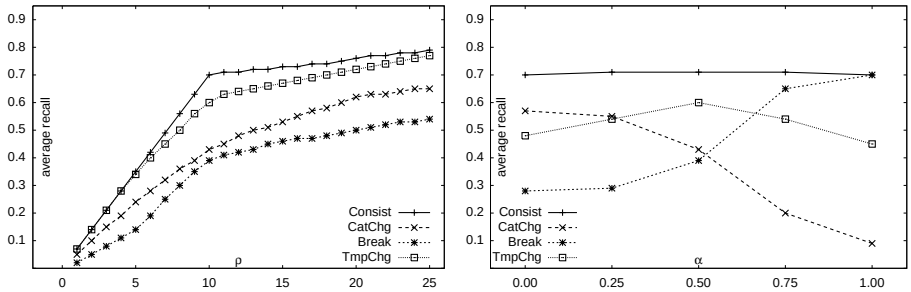

Fig. 6. Average recall across scenarios for $\alpha = 0.5$ and $\rho = 10\%$.

Temporary Changes The last scenario we investigated (*TmpChg*), targets temporary changes in a node'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 nodes 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 Figure 5. This happens because *TmpChg* can be considered as a scenario lying between an abrupt category change (*CatChg*) and publishing documents about a specific topic with small breaks (*Break*). 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 nodes publishing more relevant documents. Finally, an interesting observation emerging from this figure 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 our baseline random node selection, all methods show an increase of as much as 600% for average recall and 200% for benefit/cost ratio.

5.4 Comparison Across Scenarios

In this section, we change our experimental viewpoint, select some baseline values for α and ρ, and compare the average recall and the benefit/cost *across scenarios*.

Average Recall Analysis Figure 6 illustrates the average recall values achieved for the various publishing scenarios. When $\alpha = 0.5$ and ρ value increases up to 25% of monitored publishers (leftmost figure), we see that *Consist* achieves the highest average

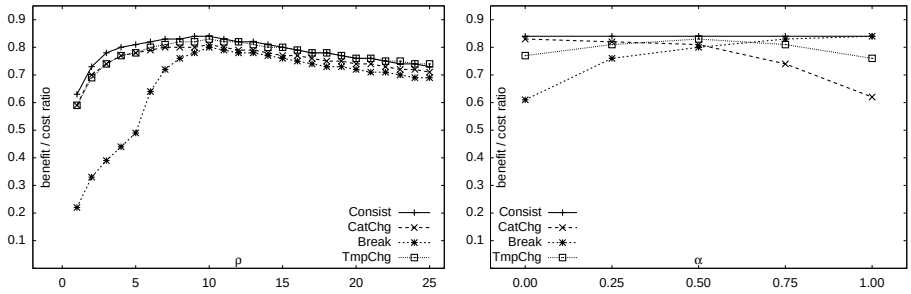

Fig. 7. Benefit / cost ratio across scenarios for $\alpha = 0.5$ and $\rho = 10\%$.

recall, since, as explained in Section 5.3, 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 a satisfactory average recall level. When ρ is set to 10% and the emphasis of the ranking method moves from publication prediction ($\alpha = 0$) to resource selection ($\alpha = 1$), average recall remains relatively unaffected for *Consist* publication scenarios (second figure). In contrast, *CatChg* and *Break* are influenced by the 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 $\alpha = 0.5$.

Message Costs Analysis The benefit/cost ratio for the different publishing scenarios is shown in Figure 7. Here, the value of ρ increases to demonstrate the benefit/cost ratio for a constant $\alpha = 0.5$ (third figure) 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% (rightmost figure).

The most important observation 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 nodes. At this value, our 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, since at least one message (the notification message) is needed at publication time to inform a subscriber about a matching document. This means that we generate an average of 0.2 extra subscription messages per notification sent.

We observe that in the *Consist* scenario, a change in the ranking method has no effect on the value of the benefit/cost ratio. 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 5.4, the highest benefit/cost ratio is achieved when combining both resource selection as node prediction scores.

6 Conclusions

We have presented the MAPS system architecture, discussed the associated protocols, and introduced a novel node selection technique based on time series analysis to support approximate IF in a P2P setting. Our experimental evaluation showed the efficiency and effectiveness of our approach in many diverse scenarios.

References

1. Yang, B., Jeh, G.: Retroactive Answering of Search Queries. WWW 2006.
2. Tang, C., Xu, Z.: pFilter: Global Information Filtering and Dissemination Using Structured Overlay Networks. FTDCS 2003.
3. Tryfonopoulos, C., Idreos, S., Koubarakis, M.: Publish/Subscribe Functionality in IR Environments using Structured Overlay Networks. SIGIR 2005.
4. Aekaterinidis, I., Triantafillou, P.: PastryStrings: A Comprehensive Content-Based Publish/Subscribe DHT Network. ICDCS 2006.
5. Tryfonopoulos, C., Zimmer, C., Weikum, G., Koubarakis, M.: Architectural Alternatives for Information Filtering in Structured Overlays. Internet Computing 2007.
6. Zimmer, C., Tryfonopoulos, C., Weikum, G.: MinervaDL: An Architecture for Information Retrieval and Filtering in Distributed Digital Libraries. ECDL '07.
7. Zimmer, C., Tryfonopoulos, C., Berberich, K., Weikum, G., Koubarakis, M.: Node Behavior Prediction for LargeScale Approximate Information Filtering. LSDS-IR 2007.
8. Terry, D., Goldberg, D., Nichols, D., Oki, B.: Continuous Queries over Append-Only Databases. SIGMOD 1992.
9. Liu, L., Pu, C., Tang, W.: Continual Queries for Internet Scale Event-Driven Information Delivery. TKDE '00.
10. Chen, J., DeWitt, D.J., Tian, F., Wang, Y.: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. SIGMOD 2000.
11. Madden, S., Shah, M.A., Hellerstein, J.M., Raman, V.: Continuously Adaptive Continuous Queries over Streams. SIGMOD '02.
12. Chandrasekaran, S., Franklin, M.J.: PSoup: A System for Streaming Queries over Streaming Data. VLDB Journal 2003.
13. Gedik, B., Liu, L.: PeerCQ: A Decentralized and Self-Configuring Peer-to-Peer Information Monitoring System. ICDCS 2003.
14. Ahmad, Y., Çetintemel, U.: Networked Query Processing for Distributed Stream-Based Applications. VLDB 2004.
15. Jain, A., Hellerstein, J.M., Ratnasamy, S., Wetherall, D.: A Wakeup Call for Internet Monitoring Systems: The Case for Distributed Triggers. HotNets 2004.
16. Zhang, R., Hu, Y.C.: HYPER: A Hybrid Approach to Efficient Content-Based Publish/Subscribe. ICDCS 2005.
17. Pietzuch, P.R., Bacon, J.: Hermes: A Distributed Event-Based Middleware Architecture. DEBS 2002.
18. Gupta, A., Sahin, O.D., Agrawal, D., Abbadi, A.E.: Meghdoot: Content-Based Publish/Subscribe over P2P Networks. Middleware 2004.
19. Ratnasamy, S., Francis, P., Handley, M., Karp, R.M., Shenker, S.: A Scalable Content-Addressable Network. SIGCOMM 2001.
20. Tryfonopoulos, C., Idreos, S., Koubarakis, M.: LibraRing: An Architecture for Distributed Digital Libraries Based on DHTs. ECDL 2005.

21. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. SIGCOMM 2001.
22. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving Collection Selection with Overlap Awareness in P2P Search Engines. SIGIR 2005.
23. Tryfonopoulos, C., Koubarakis, M., Drougas, Y.: Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators. SIGIR 2004.
24. Yan, T.W., Garcia-Molina, H.: The SIFT Information Dissemination System. TODS 1999.
25. Callan, J.: Distributed Information Retrieval. Kluwer Academic Publishers 2000.
26. Chatfield, C.: The Analysis of Time Series - An Introduction. CRC Press 2004.
27. Nottelmann, H., Fuhr, N.: Evaluating Different Methods of Estimating Retrieval Quality for Resource Selection. SIGIR 2003.