Distributed, Expressive Top-k Subscription Filtering using Covering in Publish/Subscribe Systems

Technical report

Kaiwen Zhang[1,2], Vinod Muthusamy[3], Mohammad Sadoghi[3], Hans-Arno Jacobsen[1,2]
[1]*Middleware Systems Research Group*
[2]*University of Toronto*
[3]*IBM T.J. Watson Research Center*

***Abstract*—Top-k filtering is an effective way of reducing the amount of data sent to subscribers in pub/sub applications. In this paper, we investigate top-k subscription filtering, where a publication is delivered only to the k best ranked subscribers. The naive approach to perform filtering early at the publisher edge broker works only if complete knowledge of the subscriptions is available, which is not compatible with the well-established covering optimization in publish/subscribe systems. We propose an efficient rank-cover technique to reconcile top-k subscription filtering with covering. We extend the covering model to support top-k and describe a novel algorithm for forwarding subscriptions to publishers while maintaining correctness. We also establish a framework for supporting different types of ranking semantics, such as fairness and diversity. Finally, we conduct an experiential evaluation and perform sensitivity analysis to demonstrate that our optimized rank-cover algorithm retains both covering and fairness while achieving properties advantageous to our targeted workloads.**

I. Introduction

The exponential increase in the data volume and velocity is an undisputed trend. Today, data is being generated at an astonishing rate of 2.5 billion gigabytes daily. The explosion is partly due to the Internet of Things (IoT) that increasingly connects data sources (including objects and devices) to form a complex network, one which is expected to exceed one trillion nodes by 2015 [1]. Data is now being viewed as a new natural resource in the enterprise world with an unprecedented degree of heterogeneity and countless possible ways of aggregating and consuming it [1].

Given the sheer size of the data, new challenges arise for distributed dissemination and consumption of data. Therefore, the ability to deliver only pertinent data to selected and interested consumers is an important requirement that exists in many domains ranging from social platforms to enterprise infrastructure.

In the social space, people are engaging on more platforms (e.g., Twitter and Facebook) and directly connecting to a larger audience that results in receiving a greater volume of data. Thus, the underlying system must provide a way to deliver only highly selective data to interested users. A piece of information may have a different face value for different users. The notion of *relevance* is thus essential: for instance, one person may interested in following political figures while another may value arts and theater-related content.

From the perspective of the data consumer, there is a limit on the amount of information that can be consumed before overwhelming the user. Thus, the ability to filter data is a must by allowing users to subscribe only to the data of interest. From the perspective of the data producer, there also exist limited resources in terms of communication bandwidth for data distribution. Additionally, the provider may wish to limit the size of the audience to whom data is delivered. For example, promotional offers or coupons are limited and need to be delivered only to the most important individuals, that is, those who are most likely to be interested in the offer or redeem the coupon. Selecting the matching individuals may be based on a relevance ranking along with other objectives such as *diversity*: the ability to reach out to individuals in a wide variety of demographics. Alternatively, the filtering may be motivated by *fairness*: every individual must have an equal chance of being informed about new data, irrespective of the size and properties of his or her demographic.

In the enterprise domain, the data dissemination challenges are even more pronounced: consider the ever growing size of data centers and cloud infrastructures with Google operating at the million servers scale [2]. Soon, one quarter of all the world's applications will be running on the data centers of cloud providers [1]. At this level, cloud providers offering hardware and software as services must be able to manage and distribute user-software patches (e.g., application patches, user configurations, or elastic scaling requirements) and system-specific software and security patches (e.g., OS- or VM-related patches). The situation further worsens when taking into account the complexity and size of cloud network topologies that consist of thousands of heterogeneous nodes with a wide range of required SLAs.

We are now again faced with the same challenge of efficient information dissemination (e.g., software patches) to interested consumers (e.g., servers) with well-defined ranking semantics such as relevance, diversity, and fairness. The basic notion of *relevance* is unavoidable since not every server has to be notified about every software patch, e.g., a server running Windows is not interested in Linux patches.

However, it is insufficient for operating a large-scale data center. For instance, a system administrator may want to assess the total risk of releasing a data center-wide security patch; therefore, the platform must support delivering the patch to only a representative sample of all nodes–a requirement satisfied by *diversity* delivery semantics.[1] Alternatively, important time-consuming system upgrades need to be rolled out to all nodes using a staged deployment: *fairness* must be respected when selecting the initial set of nodes. Otherwise, situations may arise such that nodes dedicated to certain clients are favored over others.

What is common in both scenarios discussed above is the existence of a complex network of interconnected consumers (i.e., subscribers) and producers of data (i.e., publishers). Furthermore, there is a rising trend to establish a decentralized architecture to cope with the data volume and velocity. The pub/sub model, known for its scalability and decoupled nature, is a logical substrate candidate for distributed data dissemination [3], [4], [5], [6], [7], [8], [9], [10], [11].

The problem addressed in this paper then is to efficiently deliver each publication to the k highest ranked matching subscriptions over a pub/sub broker network. Subscription rankings can be based on a variety of criteria including a scoring function (relevance), fairness, diversity, or a combination of these criteria. In addition, this paper considers the case where it is the publisher that regulates how widely a publication is disseminated by selecting the value of k. The complementary scheme whereby the subscriber selects k events from a window of publications has already been addressed [10].

We remark that there is a requirement for the underlying pub/sub system to employ optimized routing protocols and efficient filtering mechanisms for disseminating information flowing from data producers to data consumers. A well-known routing optimization prevalent in pub/sub systems is covering [12]. This protocol reduces the propagation and management of subscriptions that are forwarded to brokers (i.e., nodes) in the pub/sub network. Covering optimization algorithms are compatible only with simple subscription matching and do not extend trivially to more expressive semantics such as top-k filtering using relevance, fairness, or diversity.

In general, without the covering protocol, every edge broker to which a publisher is connected to must maintain global knowledge of all subscriptions in the system, thus making top-k subscription filtering trivial since the rank of all subscriptions can be computed locally. But once subscription covering is assumed (de facto in pub/sub systems), the edge broker has only a partial knowledge of the subscription space, and top-k filtering becomes a challenging and important problem.

[1]Such a semantic could be useful also for incrementally distributing software patches in a diverse manner.

The existing research in pub/sub focuses heavily on efficient routing (e.g., covering [12]) and efficient matching algorithms [3], [4], [5], [7], [8], [9] in isolation. A recent work argues for the importance of top-k matching based on the relevance of subscriptions [11], but without paying attention to the pub/sub covering aspects and without considering the extended ranking semantics such as fairness and diversity that are the main focus of our work.

In this paper, we first formalize general top-k subscription filtering semantics to realize various ranking objectives, such as relevance, diversity, and fairness. Second, we propose an efficient rank-cover algorithm to reduce the forwarded subscription traffic while satisfying the aforementioned filtering semantics. The key intuition behind our rank-cover algorithm is to construct a subscription covering structure at each broker that consists of a subset of a partially ordered set (POSET) containing only its top ranking downstream subscriptions. Therefore, our rank-cover algorithm substantially reduces the number of subscriptions forwarded upstream while guaranteeing the correctness of top-k filtering semantics at upstream brokers. We make the following contributions in this work:

- Formalize general top-k subscription filtering semantics to express a wide range of ranking objectives including relevance, diversity, and fairness.
- Develop a novel rank-cover algorithm to enable efficient distributed top-k subscription filtering, and its ancestor counting optimization to further reduce subscription traffic.
- Introduce fairness as a top-k ranking objective and develop a weighted sort shuffle algorithm for efficient subscription selection.
- Conduct an extensive sensitivity analysis of our algorithms (on an open-source pub/sub system) with respect to edge broker load reduction (covering effectiveness), end-to-end latency, processing overhead, and fairness semantics.

II. Semantics and naive solution

In this section, we first formalize a model for top-k subscription filtering in pub/sub. Our model consists of a general framework that supports a variety of ranking semantics used for selecting a top-k of subscriptions. In particular, we describe our relevance scoring, diversity, and fairness ranking criteria for top-k filtering. Second, we describe how top-k subscription filtering semantics are propagated through the system by piggybacking top-k data via advertisements. Finally, we demonstrate how covering, in conjunction with the proposed top-k model, is incompatible with a naive solution and is thus a non-trivial challenge to solve.

A. Top-k model and ranking criteria

Consider a publication p that matches a set of subscriptions S. The problem addressed in this paper is to deliver

the publication to the highest ranked k-sized subset of S. Different ranking criteria can be used to determine the subset. For example:

- *Relevance-based (scoring) semantics*. There is a scoring function, $\text{score}(p, s)$, that computes a score given a publication p and subscription s. The subscriptions are ranked by descending score, such that the subscriptions with the top-k highest scores are selected.
- *Fairness semantics*. Each rank between $[1, |S|]$ is randomly assigned with equal probability to every subscription in S. Therefore, a fair delivery selects a uniformly random k-sized subset of S. More precisely, the probability of delivering p to any subscription $s_i \in S$ is $\frac{k}{|S|}$.
- *Diversity semantics*. Suppose there is a distance metric that quantifies the pairwise similarity of subscriptions: distance(s_i, s_j). Each subscription s in S is ranked in descending order based on the value of $\sum_{j=1}^{|N|} distance(s, s_j)$, which is the cumulative sum of pairwise distances between s and all subscriptions in S. The resulting k-sized subset of S is *diverse*, the intention being to avoid delivering the publication to subscriptions with similar interests.

B. Semantics propagation

The top-k ranking semantics above are from the perspective of the publisher. In particular, we are not addressing the problem where a subscriber chooses to receive only a subset of matching publications, since it has been studied in other works [10].

As a consequence of this design point, the publisher must specify a value k with each advertisement a it issues, indicating that publications induced by a should be sent to the k highest-ranked matching subscriptions. In addition, a ranking function $rank(p, S)$ may be issued along with the subscription or advertisement, or there may be a system-wide function. In this paper, we assume the most general case of a per-subscription ranking function.

For a publication p and set of matching subscriptions S, the ranking function $rank(p, S)$ assigns a rank between $[1, |S|]$ to each subscription s in S. This ranking function can use any combination of ranking semantics described above. For example, a function could first use relevance to rank subscriptions by score. Subscriptions which are tied in score can then be further demarcated using diversity. This ranking function is used during the top-k filtering process to compute the top-k ranked subscriptions for a publication to be delivered to.

C. Naive solution

This paper focuses on the forwarding of advertisement, subscription, and publication messages to achieve top-k dissemination. Notably, the algorithm to compute the top-k set for a publication is assumed to be known. More specifically, given a set of subscriptions $\mathbb{S}$ and a publication p: (i) a matching algorithm can compute the set of subscriptions $S \in \mathbb{S}$ that match p, and (ii) a top-k algorithm can use a provided $rank(p, S)$ function to compute the k-sized set of top ranked subscriptions $\hat{S}$ among S.

If there is no subscription covering, then each broker can compute the top-k ranked subscriptions locally and forward the publications accordingly along with the IDs of the subscriptions to deliver the publication to.

In the experiments in Section IV, this algorithm which does not employ covering is referred to as REGULAR-K.

D. Problem with naive solution

The naive solution works because each broker knows the complete set of subscriptions in the system, allowing for the ranks of matching subscriptions to be computed accurately. A common pub/sub optimization is to forward only a subset of subscriptions to upstream brokers. The subset is chosen to be an appropriate summary of the subscriptions such that the upstream broker can correctly determine which downstream brokers may have matching subscriptions. For example, the subscription covering algorithm avoids forwarding subscriptions whose interest space is subsumed by another subscription [12].

A subscription s_1 covers subscription s_2 iff all publications that match s_2 also match s_1. Given a set of subscriptions, and their pairwise covering relationships, the broker constructs a partially ordered set (POSET) that represents the covering relationships among its subscriptions. It then forwards only those subscriptions that are not covered by any other.

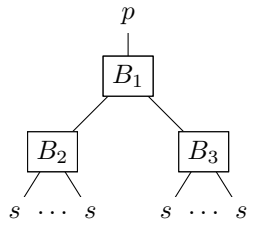


Figure 1. Simple three broker topology

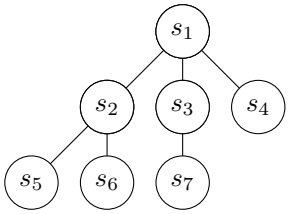


Figure 2. Covering POSET among a set of subscriptions

Consider the broker topology shown in Figure 1, where the publisher is connected to broker B_1 and the subscribers connected to brokers B_2 and B_3. Also suppose that the subscribers at broker B_2 issue the subscriptions s_1 to s_7 whose covering POSET is depicted in Figure 2. When subscription covering is not employed, broker B_1 is aware of all the subscriptions and can select the appropriate top-k set as outlined in Section II-C. However, with subscription covering, broker B_2 only forwards the roots of the POSET, subscription s_1 in this case, to the upstream broker. Broker B_1 no longer has enough information to know how many

of the top-k subscriptions are reachable from each downstream broker. In Section III, we will develop a solution to this problem that can achieve correct top-k filtering while retaining most of the benefits of subscription covering.

III. Top-k forwarding algorithms

This section presents first a distributed top-k algorithm (RANK-COVER-K) that correctly supports covering, as opposed to the REGULAR-K solution shown in Section II-C. We also present ACO-COVER-K, an optimized version of RANK-COVER-K using ancestor counting. Finally, we propose ACO-FAIR-K, a specific instance of ACO-COVER-K using our proposed fair ranker. Proofs for algorithms in this section can be found in Appendix A.

A. Top-k-aware subscription covering

We now describe a set of algorithms that support subscription covering. The challenge is to decide how many of the k most relevant subscriptions are reachable from each downstream broker.

Consider again the topology in Figure 1. It is important that brokers B_2 and B_3 only forward a minimized subset of their subscriptions to B_1, that is, they should employ covering techniques to reduce the forwarded subscription load. Suppose $k = 2$ for the publisher. Then, when a publication arrives at broker B_1, it needs to distinguish among three cases:

- The two highest ranked subscriptions are both reachable through broker B_2.
- One of the highest ranked subscriptions is reachable through broker B_2, and the other through B_3.
- The two highest ranked subscriptions are both reachable through broker B_3.

The key insight is that broker B_1 only needs to distinguish among the above three cases. In particular, it does not need to know if there are more than $k = 2$ matching subscriptions downstream of a particular broker. Therefore, it suffices for each broker to forward the k best ranked subscriptions for all possible publications that may be induced by advertisement a. Note that unlike the REGULAR-K solution, each broker along the dissemination path must perform top-k filtering to ensure that covered subscriptions can be selected.

We will now generalize and formalize this point. Let S_j be the set of subscriptions that match publication p_j at a given broker, and let $s_{j,1}, ..., s_{j,n}$ be the subscriptions in S_j ordered by ranking. Furthermore, let $\hat{S}_j$ be the top-k set in S_j, that is, $\hat{S}_j = s_{j,1}, ..., s_{j,q}$ where $q = min(n, k)$. Finally, let $\hat{S} = \bigcup_j \hat{S}_j$ be the set of highest ranking subscriptions for all possible publications. Notice that the size of $\hat{S}$ may be larger than k.

Theorem 1. *If a broker B receives the subscriptions $\hat{S}$ as computed by each of its downstream brokers, then it can compute the set of most relevant subscriptions for a given publication.*

We now outline several algorithms to compute $\hat{S}$, the set of subscriptions that each broker must forward to its upstream broker in order to allow top-k filtering with subscription covering.

1) Rank-cover: Let subscriptions $s_1, \cdots, s_n$ be the subscriptions at a given broker that intersect an advertisement a from an upstream broker.

In this algorithm, the subscription cover is a partially ordered sets (POSET) built according to an *extended covering definition* that is based on the subscription predicates and given ranking functions. We call this the **rank-cover** definition. In particular, subscription s_i rank-covers s_j iff both these conditions are true: (i) The predicates of s_i cover those of s_j. (This is the conventional covering definition.) (ii) The ranking function of s_i covers that of s_j. More precisely, for every publication p induced by advertisement a, the rank of s_i is greater than the rank of s_j.

Intuitively, rank-covering implies that if a publication matches s_j, then it will also match s_i with a higher rank.

Each broker then forwards all the subscriptions in the first d levels of the POSET. This will ensure that the upstream broker has enough subscriptions to determine how many of the top-k ranking subscriptions are downstream.

More precisely, this algorithm constructs $\hat{S}$ as $\{s_i | \text{depth}(s_i) < d\}$ where depth(s_i) is the depth of subscription s_i in the rank-cover POSET.

Theorem 2. *The set $\hat{S} = \{s_i | depth(s_i) < d\}$ contains the set of top-k subscriptions for any possible publication by advertisement a.*

For example, consider the topology in Figure 1, and let subscriptions $s_1, ..., s_7$ be the subscriptions at broker B_2 that intersect with advertisement a (which has $k = 2$) from broker B_1. Suppose also that Figure 2 depicts the covering POSET for the subscriptions at B_2 that intersect advertisement a according to the rank-cover definition. In this case, since $k = 2$, all nodes with depth less than 2 will be forwarded to B_1. So, subs s_1, s_2, s_3, s_4 will be sent to B_1. Broker B_1 now has enough subscriptions from B_2 to determine if 0, 1 or 2 of the top ranked subscriptions are from broker B_2.

Note that the algorithm must be rerun when there is a change in the set of subscriptions or advertisements, as is true of the traditional covering algorithm.

The rank-covering algorithm is defined in Algorithm 1, and is referred to as RANK-COVER-K in the experiments in Section IV.

2) Ancestor counting optimization (ACO): The rank-cover covering algorithm (Section III-A1) can further be optimized so that certain subscriptions, which have multiple parents, can be removed from the forwarding set. This

Algorithm 1: Rank cover forwarding

```
1  foreach advertisement a do
       /* Construct rank-cover POSET for subs that
          intersect a and come from a different last hop.
          */
2      S_a ← {s | s matches a ∧ s.lasthop ≠ a.lasthop}
3      C_a ← rank-covering POSET among subscriptions S_a
       /* Construct the forwarding set for a.          */
4      F ← subscriptions at depth < d in C_a
5      add F in the subs to forward to a.lasthop
```

happens if there are enough overlapping ancestors to cover the subscription and satisfy k matches.

In this optimization, rather than forwarding all subscriptions in the POSET with depth less than k, we only forward those subscriptions with fewer than k ancestors in the POSET.

More precisely, this algorithm constructs $\hat{S}$ as $\{s_i | \text{ancestors}(s_i) < d\}$ where ancestors(s_i) is the number of subscriptions in the rank-cover POSET that are in the path from the root to s_i.

Theorem 3. *The set $\hat{S} = \{s_i | ancestors(s_i) < d\}$ contains the set of top-k subscriptions for any possible publication from advertisement a.*

This algorithm is referred to as ACO-COVER-K in the experiments in Section IV.

B. Fairness

This section considers a fair ranker when selecting the k subscriptions to forward a publication. More precisely, given a publication p that matches n subscriptions $s_1, ..., s_n$, the probability of delivering p to s_i must be k/n. That is, each matching subscription is equally likely to be in the top-k set.

Note that if there is no subscription covering, then the solution is trivial as discussed in Section II-C: the first broker that receives the publication from the publisher can compute the set of matching subscriptions, uniformly select among them, and forward the ids of these selected subscriptions to each downstream broker.

One attempt to evaluate fairness on subscriptions known by the broker after applying the covering algorithms is shown in Section III-A. However, this simple-minded approach could produce incorrect results. For example, consider a broker with subscriptions S and corresponding forwarding set $\hat{S}$ as computed by the algorithms in Section III-A. Now add a new subscription s' to get $S' = \{s' \cup S'\}$ and corresponding forwarding set $\hat{S}'$. If s' is a leaf node in the predicate-covering POSET of S', and k is less than the minimum height of the rank-cover POSET of S', then s' will not be in S'. Consequently, $\hat{S} = \hat{S}'$, and hence from the upstream broker's perspective, the case where s' is present is indistinguishable from the case where it is absent, but to achieve the fairness semantic the addition of s' should affect the routing logic.

The fairness algorithm presented here is an approximate algorithm, which we argue is justifiable since the definition of fairness is itself probabilistic. In this algorithm, subscriptions are forwarded as usual, but each forwarded subscription is assigned a weight equal the number of descendants in its rank-cover POSET.

During publication forwarding, a weighted shuffle is performed among the matching subscriptions, and the first k subscriptions in the shuffled sequence are selected. The weighted shuffle algorithm, detailed in Algorithm 2 requires a search tree (eg. red-black tree), labeled as *wtree*. This tree keeps track of the cumulative sum of the weight of inserted elements for each entry. A randomly generated value in the range up to the total weight of all the entries can be used to retrieve an element whose key is the least greater to the value selected. This algorithm requires n insertions and $\mathcal{O}(n)$ searches for a total running time of $\mathcal{O}(n \log n)$.

Algorithm 2: Weighted shuffle algorithm

```
   /* Construct a cumulative weight search tree wtree:  */
1  sum ← 0
2  tail ← null
3  foreach subscription s do
4      sum ← sum + s.weight
5      insert (sum, s) in wtree
6      tail ← sum
   /* Shuffle the elements with weighted probability    */
7  slist ← {}
8  repeat
9      x ← random(1, tail)
10     e ← ceiling(x, stree)
11     slist ← slist ∪ {e}
12     replace e with a pointer to the range [tail − e.weight, tail] in stree
13     tail ← tail − e.weight
14 until |S| times
15 return slist
```

C. Discussion

The above algorithms can be optimized further if relaxed to accept approximate solutions. For example, if we control the forwarded subscription set recomputation frequency such that the recomputation rate is slower that the subscription churn rate, the number of subscription messages is reduced at the expense of an imprecise or stale top-k selection.

The framework in this paper can support diversity, but it remains to devise a reasonable and efficient pairwise distance metric among subscriptions. The definition and justification of such a metric is out of the scope of this paper, and is treated as an interesting avenue for future work.

IV. Evaluation

This section contains the results of experiments performed on our various algorithm implementations. We first compare the covering performance of three implementations: (1) our baseline top-k solution with covering (RANK-COVER-K), (2) an optimized version using ancestor counting (ACO-COVER-K), and (3) our baseline covering algorithm without top-k (COVER). These experiments gauge the impact of

using top-k over covering and provide grounds for a discussion on the sensitivity of our work with respect to various workload parameters.

We then perform an evaluation of our fair ranker using our optimized top-k solution (ACO-FAIR-K), compared to ACO-COVER-K using a fairness metric. The fairness baseline for this experiment is our top-k solution without covering (REGULAR-K). Each subscription has a fair chance to be selected in REGULAR-K since each broker has complete knowledge of all downstream matching subscriptions when covering is not employed (see Section II-C).

Finally, we measure the processing overhead of our various solutions using end-to-end publication latency to assess the impact of our top-k filtering and fairness selection algorithms on the publication match-and-forward process.

A. Setup

The implementation is performed in Java using the PADRES pub/sub prototype[2]. Experiments are performed on the SciNet testbed using the General Purpose Cluster (GPC)[3].

The workload used is synthetic, with a single publisher emitting publications at a rate of 30 per minute. A single publisher allows us to better control the distribution of subscribers relative to the publisher which is important for fairness (see Section IV-D).

The broker overlay topology consists of 5 to 10 core brokers. Each core broker is then connected to 5 edge brokers. We connect 10 subscribers to each edge broker. We employ between 250 to 500 subscribers, each with one subscription. For example, an experiment with 10 core brokers has 50 edge brokers and 500 subscribers. This setup is modeled as a network of data centers connected through gateways (core brokers) and measures the impact of our algorithms on delivery paths with multiple broker hops.

Subscriptions are randomly generated to have between 30% and 60% selectivity, uniformly distributed in the publication space. We argue that the high selectivity of the subscriptions is appropriate for our motivating scenario where users are receiving an overwhelming amount of data, hence the need for additional filtering. Furthermore, the narrow range of selectivity creates conservative covering results since neither excessively large subscriptions exist to cover large spaces, nor are small subscriptions present that can easily be covered. The relatively even size of subscriptions creates broad and shallow covering trees which constitute a worst case for our scenario. Better covering results can be achieved using deep covering trees. For the purpose of this experiment, we are interested in the worst-case results for our solutions.

Advertisements are set with a k which is between 0.2% to 2.5% of the total number of subscriptions. For instance, in an experiment with 500 subscriptions, a k set to 10 has a selectivity of 2%. Top-k is useful for suppressing an overwhelming amount of data, therefore maintaining a lower k to control the selectivity of the publications is desirable.

For all solutions except ACO-FAIR-K, we employ a covering-agnostic random ranking function. Each subscription known by the broker for a publication is assigned a random rank with uniform distribution. Since our experiments focus on fairness, this setup allows us to maximize the size of the pool of subscriptions which must be fairly selected from. Our solution is orthogonal and compatible with any ranking scheme.

The fair ranker used by ACO-FAIR-K extends the random ranker by weighing subscriptions appropriately according to the covered subscriptions (see Section III-B). Furthermore, we asynchronously update the weights for the ranker by propagating the count of covered subscriptions through propagated subscriptions (see Section III-C).

B. Metrics

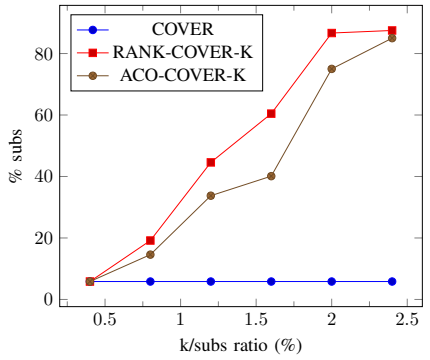
Percentage of forwarded subscriptions: Our main metric for measuring the effectiveness of our covering techniques is the reduction in the number of subscriptions sent upstream (towards publishers). In our evaluation, we measure the ratio between the number of subscriptions at the publisher edge broker and the total number of subscriptions sent by all subscribers. Since the publisher edge broker is the last hop for every subscription in the system (as they all have non-zero selectivity), this number is an overall measure for covering in the system. The lower bound is provided by our baseline covering algorithm COVER which does not support top-k, while the upper bound is 100% for top-k without covering (REGULAR-K). Our proposed solutions fall somewhere in between and allow us to assess the impact of top-k dissemination on covering.

End-to-end latency: We measure the publication latency from the time elapsed between its sending at the publisher to the receipt at a subscriber. This allows us to measure the aggregated processing overhead of our top-k algorithm being run at every broker hop encountered. In particular, this allows us to assess the impact of the weighted fair shuffle algorithm developed in this paper.

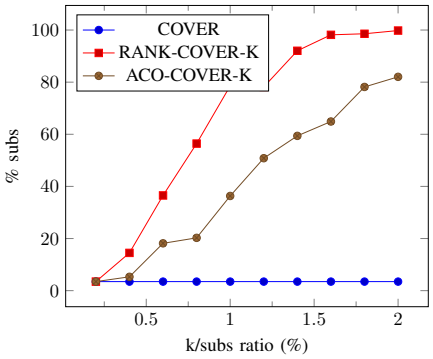
Fairness: We measure fairness empirically by comparing the number of publications each subscription receive for each implementation to the baseline top-k algorithm REGULAR-K. We sum up the difference in number of publications received for all the subscriptions during the course of the experiment. This number indicates how much deviation the implementation has to the expected number of publications to be received from each subscription. Note that the subscriptions and publications are generated deterministically for every experiment.

[2]http://padres.msrg.toronto.edu/

[3]http://www.scinet.utoronto.ca/



(a) 250 subscribers



(b) 500 subscribers

Figure 3. Covering effectiveness

C. Covering performance

Figures 3(a) and 3(b) show the covering performance of our various algorithms relative to the baseline COVER with varying k values and different amount of subscriptions. A lower number indicates that the covering algorithm has been more effective in reducing the number of subscriptions disseminated. The graphs show that our solutions, RANK-COVER-K and ACO-COVER-K, are sensitive to increasing values of k. For example, we note that covering is completely nullified (no reduction in subscriptions, 99.8% subscriptions) for RANK-COVER-K at k set to 2% ($k = 10$) for 500 subscriptions. As described in Section IV-A, we selected subscriptions such that the covering trees are shallow and broad. Therefore, traversing a few levels deep is enough to encounter a large number of subscriptions. However, we argue that for our intended applications, low values of k (e.g., less than 1%) are appropriate to justify top-k filtering.

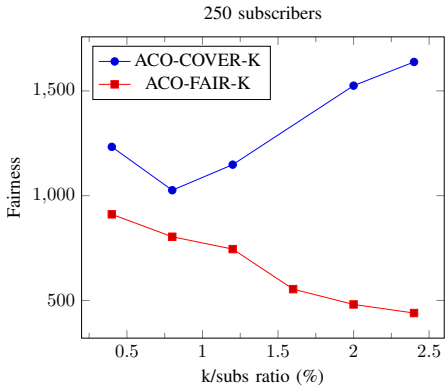
.

We also notice that the baseline propagates only 5.83% of 250 subscriptions, and 3.47% for 500 subscribers. Note that the baseline performance is unaffected by the value of k, since it does not support top-k. COVER therefore demonstrates scalability through the improvement in subscription reduction with increasing subscription loads. RANK-COVER-K does not preserve this trend, as the covering effectiveness decreases for the same relative k in a workload with more subscriptions. As an example, 86.67% of subscriptions are present for 250 subscriptions which rises to 99.8% for 500 subscriptions when the $k/subs$ ratio is set to 2%. On the other hand, ACO-COVER-K is less sensitive to increasing subscriptions: at the same $k/subs$ ratio of 2%, the % of subscriptions slightly increases from 75% to 82.04%. Since the optimization leverages the presence of subscriptions with multiple parents to cover subscriptions more effectively, we can infer that the incidence of multiple parents increases as the number of subscriptions increase, which is consistent with the selectivity parameters described in the setup above. Increasing the number of subscriptions does not necessarily increase the depth of the covering tree, but the breadth of existing levels, which is useful for ACO-COVER-K but not for RANK-COVER-K.

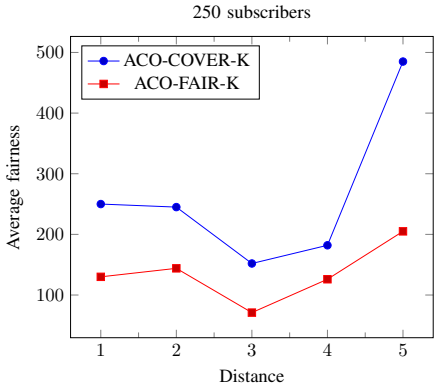
Summary: Our covering top-k solutions are sensitive to varying values of k. RANK-COVER-K also does not scale to the number of subscriptions: at 500 subscriptions, RANK-COVER-K does not reduce the number of subscriptions for k set to 2% (10). These properties are advantageous to ACO-COVER-K: top-k is employed to enforce a low rate of selectivity, hence does not require a large k, and filtering on subscriptions is performed in workloads with a large number of subscriptions, thus requiring scalability in that aspect.

D. Fairness evaluation

Figure 4(a) evaluates the fairness of our optimized solution ACO-COVER-K to ACO-FAIR-K, which is the same optimized solution equipped with our fair ranker. A lower value of fairness is better, as it indicates that the solution sends a number of publications to each subscription closer to the expected value established by REGULAR-K. We note that our fair ranker outperforms the random ranker at all values of k tested by an average of 224%. ACO-FAIR-K also becomes increasingly fair as the value of k increases. This is due to the increase in notification volume which comes with larger values of k. With more subscriptions being selected, the fair ranker's ability to do a weighted shuffle of those subscriptions is better leveraged. Furthermore, increasing the k value triggers more subscriptions to be sent upstream. Since the metadata used by the fair ranker is propagated via subscriptions, this increased frequency in subscription propagation increases the accuracy of the metadata. On the other hand, ACO-COVER-K decreases in fairness as more brokers are encountered as the random ranker used further impacts the fairness of the solution. We also note that the fairness of ACO-COVER-K improves by 20.18% between $k = 0.4\%(1)$ to $k = 0.8\%(2)$. As k increases, the number of subscriptions which are uncovered rises by 13.34% (as seen in Figure 3(a)). This larger number of subscriptions found at the publisher edge broker benefits the random ranker which now has a more accurate representation of the subscriptions in the system, but this benefit is not enough to offset the



(a) Overall fairness



(b) Fairness by distance

Figure 4. Fairness evaluation

inaccuracy of larger selections with higher values of k.

In light of these observations, we conclude that ACO-FAIR-K outperforms ACO-COVER-K significantly and does not suffer from issues that arise when the k value is at the extreme ends of the intended range for k.

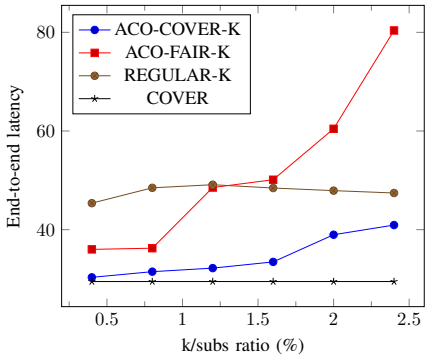
Figure 4(b) shows the sensitivity of fairness for subscriptions at varying distance from the publisher. Distance is measured as the number of core broker hops between the publisher and the subscriber. The figure shows the average fairness of subscriptions across the different values of k, since the same pattern is observed regardless of the k used. We observe that for both solutions, fairness improves when it reaches 3 broker hops, and then decreases when the number of hops increases beyond. This can be explained due to the nature of the inaccuracies. The subscriptions closer are more numerous at the publisher edge broker: because they are propagated through less brokers to reach the publisher, there are less opportunities for them to be covered. On the other hand, subscriptions farther away are less likely to reach the publisher, since there are more opportunities for them to be covered by other subscriptions in intermediary brokers. As a result, subscriptions closer are overvalued and receive more publications, while subscriptions further away are undervalued and receive less publications. In both cases, the fairness metric increases as these subscriptions deviate from the expected outcome. ACO-FAIR-K is subject to this trend to a lesser degree, but for a different reason. The fairness ranker relies on the meta-information which is propagated through subscriptions. In the workload used, staleness in the meta-information tends to underestimate the number of subscriptions. Because these inaccuracies accumulate over multiple hops, subscribers which are closer to the publisher tend to be overvalued, while ones farther away are undervalued. Therefore, we observe the discrepancy in fairness for the closest subscriptions because they receive more publications than expected, while the subscriptions further away loses fairness as they receive less publications than expected.

Summary: ACO-FAIR-K, which uses our fair ranker, outperforms ACO-COVER-K, which selects randomly from known subscriptions, by an average of 224% in the target range for k and becomes increasingly better as the value of k increases. Both solutions tend to send publications to closer subscriptions more frequently, while subscriptions further away are less likely to be selected. ACO-FAIR-K is less sensitive in that regard. Additionally, the diameter of pub/sub overlays can be controlled to lessen this effect [13].

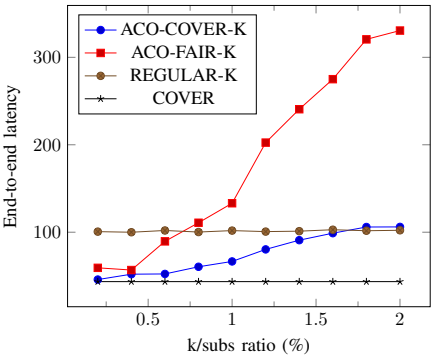
E. Processing overhead

Figures 5(a) and 5(b) show the end-to-end latency of publication delivery for 250 and 500 subscriptions, respectively. We first note that REGULAR-K is outperformed by ACO-COVER-K in lower values of k. This indicates that the processing overhead for top-k is affected by the number of subscriptions stored at a broker. As the value of k grows larger, the covering effectiveness (as explained in Section IV-C) decreases which affects the top-k overhead. Using values extracted from the COVER and ACO-COVER-K results, we evaluate the overhead at a broker of top-k processing to be 3% of the total latency, if the same number of subscriptions is stored at the broker. The most significant part of the top-k processing overhead is therefore due to the drop in covering performance, which increases the number of subscriptions residing at each broker, which in turn affects the matching and processing times.

The overhead of the fair sorter has a significant impact on latency in higher values of k tested. The overhead follows a superlinear growth to the number of subscriptions, as observed in Figures 6(a) and 6(b). This is in line with the $\mathcal{O}(n \log n)$ running time of our weighted shuffle implementation which is invoked at each broker when processing a publication. This result suggests that adaptive solutions which can either limit the number of subscriptions to be shuffled or can avoid the shuffle selection at certain brokers will improve publication latency. It also denotes the importance of our optimization for the weighted shuffle procedure. Nevertheless, the experiments indicate that for values of

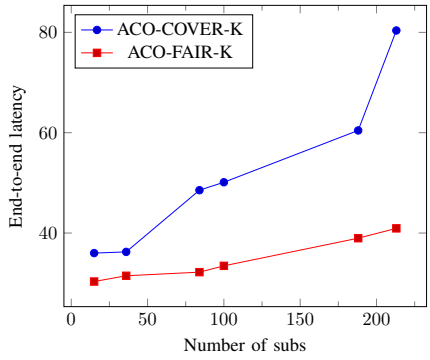


(a) 250 subscribers

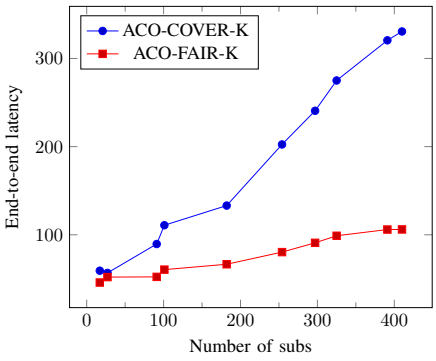


(b) 500 subscribers

Figure 5. End-to-end latency



(a) 250 subscribers



(b) 500 subscribers

Figure 6. Latency sensitivity

k less than 1% of the total number of subscriptions, it is possible to maintain fairness and still perform better than a top-k implementation with no covering.

Summary: Top-k covering is shown to improve end-to-end latency by an average of 25% in the range of k tested, with the benefit lessening as the covering effectiveness decreases. The fair ranker has a significant impact on the processing overhead at a broker which is sensitive to the number of subscriptions processed. This denotes the importance of optimizing the weighted shuffle procedure, which is performed at every broker while processing publications.

V. Related Work

Generally speaking, problems related to retrieving the most relevant answers have been studied in different contexts including database (distributed) top-k querying ([14], [15], [16], [17], [18]) and publish/subscribe (pub/sub) matching techniques [3], [4], [5], [6], [7], [8], [9], [10], [11].

The most widely adopted database top-k processing model [14], [15] differs with our proposed top-k model in an important respect: Our top-k model solves the reverse problem. In the database context, top-k querying means finding the most relevant tuples (events) for a given query (subscription). But in our pub/sub abstraction, matching means finding the relevant subscriptions (queries) for a given event (tuple).

Broadly speaking, two classes of matching algorithms have been proposed for pub/sub: counting-based [3], [5], [7] and tree-based [4], [6], [9], [19] approaches. A fresh look at enhancing pub/sub matching algorithms is to leverage top-k processing techniques to improve matching performance. An early top-k model is presented in [20]; however, this model leverages a fixed and predetermined scoring function, i.e., the score for each expression is computed independent of the incoming event. In addition, this approach is an extension of R-Tree, the interval tree, or the segment tree structure; as a result, it is ideal for data with few dimensions [20]. In contrast, a scalable top-k model that supports up to thousands of dimensions while incorporating a generic scoring function, i.e., takes the event into consideration, is introduced in [7], which relies on a static and single-layered pruning structure. To alleviate these challenges, a new dynamic and multi-layered pruning top-k structure is developed in [19]. However, our proposed top-k model attempts to solve a different problem, namely, distributed top-k processing; therefore, our model can leverage any of the existing top-k work as building blocks (e.g. [20], [7], [19]).

Another important aspect of pub/sub top-k matching is to explore and identify a plausible top-k semantics. Unlike in the database context, formalizing top-k semantics in pub/sub is more involved and not limited to a single

interpretation [21], [22], [23]. The most widely used pub/sub top-k semantics is defined with respect to subscribers, i.e., consumer-centric semantics, in which the subscription language is extended (with a scoring function) in order to rank each incoming publication (over a time- or count-based sliding window); thus, delivering only the top-k matched publications to each subscriber [21], [22], [23].

Alternatively, the top-k semantics can be defined with respect to a publisher, i.e., producer-centric semantics, which extends the publication language for ranking subscribers and delivering publications only to the top-k matched subscribers [20], [7]. Producer-centric semantics is suitable for targeted advertisement (e.g., targeting a specific demographic group) and diversified advertisement (e.g., reaching out to most eligible members within each demographic group). Finally, hybrid semantics can be foreseen such that both subscribers and publishers have control on how data is received and disseminated, respectively.

Recent works argue for the importance of top-k matching based on the relevance of subscriptions (i.e., producer-centric) using a non-monotonic and dynamic scoring [11], but without paying attention to the covering routing techniques employed by pub/sub systems and other top-k ranking semantics such as fairness and diversity, which is the focus of our work. A window-based top-k matching solution that also considered covering was studied in [10], but unlike our present work, their focus was on consumer-centric filtering.

In the present work, we focus primarily on producer-centric semantics that is publisher-driven and enables ranking the consumers or subscribers. More importantly, we further enrich the producer-centric matching semantics to satisfy ranking semantics beyond basic relevance scoring and to include fairness and diversity while addressing the subscription covering challenges that arise.

VI. Conclusions

In this paper, we address the problem of subscription filtering with covering in light of our extended top-k semantics that supports general ranking objectives, such as relevance, fairness, and diversity. In particular, we introduce a novel rank-cover algorithm to construct an efficient covering POSET that respects our extended top-k semantics. We further develop various optimizations to effectively prune and reduce the cover size. We conclude our work with an extensive set of sensitivity analyses of our algorithms incorporated in PADRES, an open-source publish/subscribe system, that illustrate the importance of rank-cover with respect to end-to-end latency and fairness.

References

[1] "The IBM strategy," http://www.ibm.com/annualreport/2013/, 2013.

[2] J. Lee, "The worlds largest data centers: Their facts and stats," http://www.business2community.com/infographics/the-worlds-largest-data-centers-their-facts-and-stats-0554302, 2013.

[3] T. Yan and H. Garcia-molina, "Index structures for selective dissemination of information under the boolean model," *TODS'94*.

[4] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching events in a content-based subscription system," in *PODC'99*.

[5] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, "Filtering algorithms and implementation for fast pub/sub systems," *SIGMOD'01*.

[6] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith, "Efficient filtering in publish-subscribe systems using binary decision diagrams," in *ICSE'01*.

[7] S. Whang, C. Brower, J. Shanmugasundaram, S. Vassilvitskii, E. Vee, R. Yerneni, and H. Garcia-Molina, "Indexing boolean expressions," in *VLDB'09*.

[8] M. Fontoura, S. Sadanandan, J. Shanmugasundaram, S. Vassilvitski, E. Vee, S. Venkatesan, and J. Zien, "Efficiently evaluating complex Boolean expressions," in *SIGMOD'10*.

[9] M. Sadoghi and H.-A. Jacobsen, "BE-Tree: An index structure to efficiently match Boolean expressions over high-dimensional discrete space," in *SIGMOD'11*.

[10] K. Zhang, M. Sadoghi, V. Muthusamy, and H. Jacobsen, "Distributed ranked data dissemination in social networks," in *ICDCS'13*.

[11] W. Culhane, K. R. Jayaram, and P. Eugster, "Fast, expressive top-k matching," in *Middleware '14*.

[12] G. Mühl, "Large-scale content-based publish-subscribe systems," Ph.D. dissertation, TU-Darmstadt, 2002.

[13] M. Onus and A. W. Richa, "Minimum maximum-degree publish-subscribe overlay network design," *TON'11*.

[14] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *PODS'01*.

[15] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *Comput. Surv.'08*.

[16] B. Babcock and C. Olston, "Distributed top-k monitoring," in *SIGMOD'03*.

[17] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," in *PODC'04*.

[18] S. Michel, P. Triantafillou, and G. Weikum, "Klee: a framework for distributed top-k query algorithms," in *VLDB'05*.

[19] M. Sadoghi and H.-A. Jacobsen, "Relevance matters: Capitalizing on less (top-k matching in publish/subscribe)," in *ICDE'12*.

[20] A. Machanavajjhala, E. Vee, M. Garofalakis, and J. Shanmugasundaram, "Scalable ranked publish/subscribe," *VLDB'08*.

[21] K. Pripužić, I. P. Žarko, and K. Aberer, "Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w," in *DEBS'08*.

[22] M. Drosou, E. Pitoura, and K. Stefanidis, "Preferential publish/subscribe," in *PersDB'08*.

[23] M. Drosou, K. Stefanidis, and E. Pitoura, "Preference-aware publish/subscribe delivery with diversity," in *DEBS'09*.

APPENDIX

[Proofs for Section III]

Theorem 1. *If a broker B receives the subscriptions $\hat{S}$ as computed by each of its downstream brokers, then it can compute the set of most relevant subscriptions for a given publication.*

Proof: Suppose by way of contradiction that there exists a subscription $s' \notin \hat{S}$ that is in the top-k set of subscriptions for a given publication. Consider the neighboring downstream broker B' where s' came from. Since s' is in the top-k set in S, and since the set of subscriptions S' at broker B' is a subset of S, then s' must be in the top-k set in S', that is $S' \in \hat{S}'$. If this is the case, however, then s' will be forwarded to the upstream broker and $s' \in \hat{S}$. This contradicts the original supposition. ■

Theorem 2. *The set $\hat{S} = \{s_i | \textit{depth}(s_i) < d\}$ contains the set of top-k subscriptions for any possible publication by advertisement a.*

Proof: Suppose by way of contradiction that there exists a subscription $s' \notin \hat{S}$ that is among the top-k highest ranking subscriptions for some publication. Consider the set of subscriptions $\tilde{S}$ in the path from the root of the POSET to s'. By our supposition, there are at least k such subscriptions in $\tilde{S}$. Also by the definition of the POSET construction, for any publication that matches s', each subscription in $\tilde{S}$ will also match p and have a higher score than s'. Therefore s' cannot be among the top-k most relevant subscriptions for p. ■

Theorem 3. *The set $\hat{S} = \{s_i | \textit{ancestors}(s_i) < d\}$ contains the set of top-k subscriptions for any possible publication from advertisement a.*

Proof: The proof is almost identical to that of Theorem 2. ■