Top-k/w Publish/Subscribe: Finding k Most Relevant Publications in Sliding Time Window w*

Krešimir Pripužić†
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3
Zagreb, Croatia
kresimir.pripuzic@fer.hr

Ivana Podnar Žarko
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3
Zagreb, Croatia
ivana.podnar@fer.hr

Karl Aberer
School for Computer and Communication Science
Ecole Polytechnique Fédérale de Lausanne
EPFL-IC-IIF-LSIR, Bâtiment BC, Station 14
Lausanne, Switzerland
karl.aberer@epfl.ch

ABSTRACT

Existing content-based publish/subscribe systems are designed assuming that all matching publications are equally relevant to a subscription. As we cannot know in advance the distribution of publication content, the following two unwanted situations are highly possible: a subscriber either receives too many or only few publications. In this paper we present a new publish/subscribe model which is based on the *sliding window computation model*. Our model assumes that publications have different relevance to a subscription. In the model, a subscriber receives k most relevant publications published within a time window w, where k and w are parameters defined per each subscription. As a consequence, the arrival rate of incoming relevant publications per subscription is predefined, and does not depend on the publication rate. Since all relevant objects (i.e. publications in our case) cannot be kept in main memory, existing solutions immediately discard less relevant objects, and store only a small representative set for subsequent delivery. In this paper we develop a *probabilistic criterion* to decide upon the arrival of a new object whether it may become the top-k object at some future point in time and should thus be stored in a special publications queue. We show that by accepting typically very small probability of error, the queue length is reasonably small and does not significantly depend on publication rate. Furthermore, we experimentally evaluate our approach to demonstrate its applicability in practice.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Communications Applications; H.2 [**Database Management**]: Systems—*Query processing*

General Terms

Algorithms, Experimentation, Performance

1. INTRODUCTION

Despite of extensive research efforts done in the last 15 years, content-based publish/subscribe (CBPS) systems are yet to be widely deployed. According to [33], the main reasons for their slow acceptance are the following: 1) complexity of the general CBPS problem, 2) system heterogeneity and 3) lack of wide-area deployments of CBPS systems. In this paper we argue that there is another reason for the lack of adoption of CBPS systems—an unpredictable number of matching publications that are delivered to subscribers. Either too many or only few received publications may cause user dissatisfaction with a provided service, for example, in applications such as RSS news feeds, network monitoring, or advertisement dissemination. Moreover, in networks with limited resources such as MANETs or sensor networks, it is highly desirable to minimize and control network traffic.

1.1 Motivation

In current systems, subscription is a stateless Boolean function [28]: A decision whether to deliver a publication to a subscriber is made based on the result of the matching process comparing the publication and subscriber's subscription as shown in Figure 1. The matching process depends only on the publication and subscription content, and does not take into account any additional information present in the system. This approach has the following drawbacks:

- subscriber may receive only few publications;
- subscriber may be flooded with too many publications;
- there is no ranking function to compare publications; and

*This work has been supported by the Ministry of Science, Education and Sports of the Republic of Croatia within the research project "Content Delivery and Mobility of Users and Services in New Generation Networks", and by the Federal Commission for Scholarships for Foreign Students of the Swiss Government.

†The author spent the last academic year at Ecole Polytechnique Fédérale de Lausanne as scholarship holder of the Swiss Scholarship for University Studies, Fine Arts and Music Schools.

DEBS '08, July 1-4, 2008, Rome, Italy

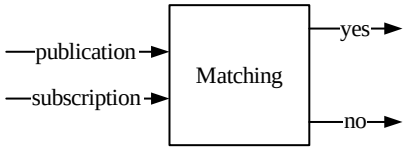

Figure 1: Matching in current systems

- partial matching between subscriptions and publications is not supported.

As the actual distribution of publication content is in general unknown in advance, it is impossible to predict the number of future publications matching an existing subscription. If a subscription is *too general*, a subscriber may receive too many publications. On the contrary, in case of an *over-specified* subscription, the subscriber may receive too few publications or none in the worst case. Thus, a subscriber has to specify an "ideal" subscription to receive an optimal number of matching publications. It is a sort of guessing, where even a slight change in subscription results in a drastically different number of matching publications. In general, an end user percepts the entire system through both the quantity and quality of received publications. Therefore, a large quantity of received publications will be considered as a sort of spam, while a system that delivers few publications might be recognized as non-working. The number of received publications is crucial for the acceptance of an actual system by end users even more if, for example, subscribes pay for each delivered publication matching subscriber information interest.

The quality of a publication can be assessed through its relevance to a subscription. In information retrieval, relevance is the measure of probability that an object will satisfy a given query [25]. Similarly, for publish/subscribe systems relevance can be defined as the measure of probability that a publication satisfies a given subscription. If we assume the content of every publication is unique (in its present context), different publications have different relevance to subscriptions. State-of-the-art publish/subscribe systems do not support publication ranking and all matching publications are considered equally relevant to a subscription. One one side, without publication ranking it is impossible to avoid delivery of a large number of matching publications in case subscription is too general. On the other side, over-specified subscriptions may deliver too few publications unless partial matching is supported. Unfortunately, as a consequence of the complexity of the general CBPS problem, state-of-the-art systems do not support partial matching and focus on fast matching of publications to subscriptions [18].

1.2 Proposed Solution

In this paper we present a new publish/subscribe model that ranks publications according to their relevance to a subscription and delivers top-k publications per subscription in a predefined sliding time window. Therefore, the quantity of received publications does not depend on the number of published publications. Obviously, the quality of received publications will depend on the relevance of published publications to a defined subscription, but statistically the quality will probably be proportional to number of published publications. Therefore, matching in our model is based on both

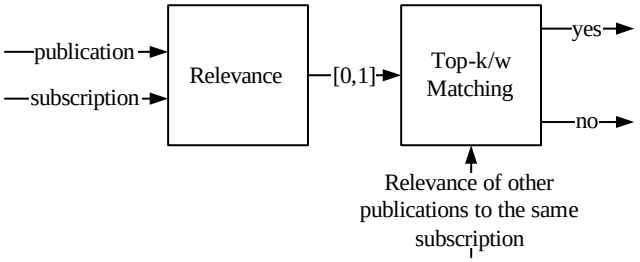

Figure 2: Matching in our model

relevance of a publication to a subscription and the relevance of other publications to the same subscription as depicted in Figure 2.

For any subscription, it is possible to deterministically identify k most relevant publications published in the past (e.g. top-5). The problem is known as the top-k problem in information retrieval [26]. The top-k/w publish/subscribe is a bit different as we have to identify most relevant publications among those published in both past and future. Obviously, this cannot be done deterministically in advance since each publication is in a way competing with nonexistent publications. However, we can compare the relevance of a newly published publication to a certain number of the most relevant previously published publications, but it is not clear how many of them should we take into account. We propose that a subscriber defines the number of publications competing simultaneously for a position among the top-k publications either by defining the number of competing publications (e.g. top-12/350) or the time window (e.g. top-8/day). We refer to the former as *number-based window* and the latter as *time-based window*. The time window does not depend on the intensity of publishing and is therefore used in our approach.

A practical solution to the top-k/w (i.e. top-k relevant publications in a time window w) problem is based on the existence of a sorted set of previously published publications for each subscription, to which we will refer as *publications queue*. Using this queue, it is possible to determine for each newly published publication, at a point in time of its publishing, whether it is among top-k publications in the queue similarly to the sliding window computation model [14]. As time passes, the newly published publications enter the queue, and the ones older than the size of the time window are dropped from it. At some later point in time, it is possible that a publication, which was not among top-k publications in the queue at the moment of its publishing, becomes a top-k publication in the queue, because some more relevant and older publications can be dropped from the queue, while younger publications become less relevant than the observed publication.

For example, let us take a look at Figure 3. At each point in time a publication older than $w = 5$ (the crossed one) is dropped from the queue, while a new one (the gray one) is added to it. At $t = 6$ the publication published at $t = 1$ is dropped, and a new one is added to the queue. The publication born at $t = 5$ becomes a top-2 publication at $t = 8$ although it was not among top-2 publications in the queue at the moment of its publication.

Thereby, two possible sets of top-k/w publications exist: 1) the set of publications that are among top-k publications

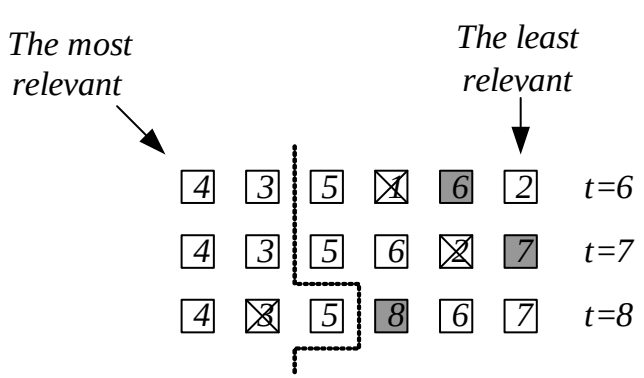

Figure 3: An example for $k = 2$ and number-based window $w = 5$

in the queue at the moment of their publishing, and 2) the set of publications that are among top-k publications in the queue at a moment of their publishing or will be in some later point in time, before time window w passes. Without the loss of generality, we will focus on the latter set. This causes a problem related to the length of publications queue because shorter queue requires less processing and less memory. Intuitively, it is clear that some publications will unlikely become a top-k publication in the queue at some later point in time. Therefore, by dropping such publications at the moment of their publishing, we can reduce the required length of the queue.

We use the relevance of the last publication (i.e. threshold) in a queue as the criterion for keeping newly published publications in the queue and accept a predefined probability of error σ for making an erroneous decision of dropping a publication that will become a top-k publication in a future time window. The sum of probabilities to become a top-k publication before time w passes has to be smaller than σ. Within this setup, we determine the required length of publications queue that certainly contains all publications with more than a minimal probability σ of becoming a top-k publication before time w passes. Our approach is purely probabilistic and is based on the assumption that relevances of publications to a subscription are independent of their publication time. For each subscription, publications queue length can be calculated at a point when subscription is activated. The length does not depend on publication intensity and can be determined a priori for each subscription.

1.3 Real World Example

Suppose that there exists a popular commercial website for renting and selling apartments where each publication is an add that defines a few attributes describing an advertised apartment (e.g. the number of rooms, size, floor, location, price, etc.). In state-of-the-art CBPS systems, a subscription would define characteristics of a desired apartment and thus it would cover a subspace of the given attribute space. As a consequence, such subscription may cause the drawbacks mentioned in Section 1.1. Suppose that we take another approach and define subscriptions as attribute space points, such that each subscription describes subscriber's "ideal" apartment. Then, using some user defined ranking function, the system could achieve ranking of published adds according to a given subscription. Moreover, using our model, a subscriber could subscribe to certain number of most related adds per day, and therefore receive the predictable number of adds per each day.

1.4 Contributions

To summarize, we make the following contributions in the paper: We propose a novel CBPS model that uses relevance between publications and subscriptions as the matching criterion and is based on sliding window computation model to choose top-k publications within sliding window w. The practical implementation of the model requires a publications queue for which we prove that its length does not depend on publication rate while accepting a predefined probability of error. Experiential results show that our model is applicable in practice.

The remainder of the paper is structured in the following way. We formally define our model in Section 2 and address the problem of publications queue length in Section 3. We experimentally evaluate our approach in Section 4 using both the representation of publications and subscriptions, and datasets from [4]. An overview of related work is presented in Section 5, and we give our conclusion and directions for future work in Section 6.

2. TOP-K/W MODEL

In this section we formally define a new publish/subscribe model called *top-k/w publish/subscribe model*. In this model, a subscriber receives only those publications that are among k most relevant ones published in a time window w, where k and w are constants defined per each subscription. We can interpret this situation in the following way. For a subscription s there exists a sliding time window $w_s(t)$ that shifts through time. It starts from a point in time when s is activated, and continues until a point in time in which s expires. If a publication is among top-k publications in any of these different window positions it will be delivered to the subscriber of s. It is important to notice that the time window has different sizes at different positions in time, as shown in Figure 4.

As a consequence, the arrival rate of relevant publications received per subscription is predictable and does not depend on the publishing rate of publications. Publications that are among top-k publications at the moment of their publishing, to which we will refer as *excellent candidates* will be delivered to the subscriber immediately after being published. Other relevant publications, i.e. such publications that become the top-k publications at some later point in time, to which we will refer as *good candidates*, will at that later point be delivered to the subscriber. Hereafter we define the model formally.

We define a triple $\mathcal{B} = (\mathcal{C}, \mathcal{P}, \mathcal{S})$, where $\mathcal{C}$ is a finite set of clients, $\mathcal{P}$ is a finite set of publications, and $\mathcal{S}$ is a finite set of subscriptions in a system. A client $c \in \mathcal{C}$ may publish publications from $\mathcal{P}$, or it may activate subscriptions from $\mathcal{S}$, or both. A client $c \in \mathcal{C}$ that activated a subscription $s \in \mathcal{S}$ is called the subscriber of s. Analogously, a client $d \in \mathcal{C}$ that published a publication $p \in \mathcal{P}$ is called the publisher of p.

Definition *(Publication)* Suppose that t_p and τ_p are two points in time such that $t_p < \tau_p$, and $c \in \mathcal{C}$ is a client in the system. We define publication $p \in \mathcal{P}$ as some content[1]that is published by c in t_p and expires in τ_p.

We say that a publication $p \in \mathcal{P}$ is *active* at a point in time t, if it is published before $t+\delta_p$, and will expire at some later point in time, where δ_p is the *maximal publication diffusion delay* in the system. Formally, $t_p + \delta_p \leq t \wedge \tau_p \geq t$.

Definition *(Active Publications)* Suppose that t is a point in time. We define a set of all active publications $P^A(t) \subseteq \mathcal{P}$ in the system at t as:

$$P^A(t) \stackrel{\text{def}}{=} \{p \in \mathcal{P} : t_p + \delta_p \leq t \wedge \tau_p \geq t\}. \quad (1)$$

It is important to notice that the set $P^A(t)$ depends on the parameter t. Therefore, at different points in time, a different set $P^A(t)$ of currently active publications exists.

Definition *(Subscription)* Suppose that t_s and τ_s are two points in time for which $t_s < \tau_s$, and $c \in \mathcal{C}$ is a client in the system. We define subscription $s \in \mathcal{S}$ as some data[1] about the interest of c that is activated in t_s and expires in τ_s. For every subscription, the following two parameters are defined[2]: an integer k_s and a real number w_s.

Analogously to active publications, a subscription $s \in \mathcal{S}$ is *active* in t if it is activated before $t + \delta_s$ and will expire after t:

$$S^A(t) \stackrel{\text{def}}{=} \{s \in \mathcal{S} : t_s + \delta_s \leq t \wedge \tau_s \geq t\}, \quad (2)$$

where δ_s is the *maximal subscription processing delay* in the system.

Definition *(Candidate Publications)* Suppose that $c \in \mathcal{C}$ is a client in the system, $s_c \in \mathcal{S}$ is a subscription of c, and t is a point in time in which s is active. We define the set of *candidate publications* $P_s^C(t) \subseteq P^A(t)$ for s_c in t as

$$P_s^C(t) \stackrel{\text{def}}{=} \{p \in P^A(t) : t_p > \max(t_s, t - w_s)\}. \quad (3)$$

Candidate publications at t are all publications published after subscription activation at t_s, but only those published within the preceding time window w_s, i.e. $t_p > t - w_s$. As the set $P_s^C(t)$ depends on both s and t, note that for a subscription $s \in \mathcal{S}$ at different points in time t for which s is active, different sets $P_s^C(t)$ exist.

At this point it is important to note that each publication is a candidate publication for all active subscriptions at the moment of its publishing. Therefore, a set of candidate publications has to be additionally filtered before delivery to subscriber. Otherwise, all subscribers with at least a single subscription would receive all publications published in the system. The following function performs such filtering.

Hypothesis *(Relevance)* Suppose that for $\mathcal{P}$ and $\mathcal{S}$ the following function exists[3]:

$$relevance : (\mathcal{P} \times \mathcal{S}) \mapsto [0, 1], \quad (4)$$

where $\forall p \in \mathcal{P}, \forall s \in \mathcal{S} : \nexists r \in \mathcal{P} \wedge p \neq r \wedge relevance(p, s) = relevance(r, s)$.

[1]To simplify the discussion in this section, we will neglect the actual representation (i.e. content and structure) of publications and subscriptions. A reader interested in the representation should refer to Section 5.

[2]The meaning of these parameters is explained in Section 1.2, and will be formally defined in the rest of this section.

[3]This function is differently defined for various representations of subscriptions and publications in different domains of interest.

In other words, each publication $p \in \mathcal{P}$ is relevant to a subscription $s \in \mathcal{S}$ with an unique degree of relevance between 0 and 1. The degree of relevance 0 means that a publication is not relevant to subscription, and 1 means that the publication is completely relevant to subscription. This function *breaks ties* such that two different publications with an equal degree of relevance to the same subscription do not exist.

Using (4) as a binary relation for comparing the relevance of publications from $\mathcal{P}$ to a subscription $s \in \mathcal{S}$, we can actually define an order on $\mathcal{P}$.

Definition *(Order)* For every subscription $s \in \mathcal{S}$ we define a binary relation $<_s$ over $\mathcal{P}$ as follows:

$$<_s \stackrel{\text{def}}{=} \{(p, r) : p, r \in \mathcal{P} \wedge [relevance(p, s) < relevance(r, s)]\}. \quad (5)$$

An order $<_s$ will be a *strict total order* on $\mathcal{P}$ if it is irreflexive, transitive and total. These are requirements on both function (4) and order (5). Therefore, for an actual domain of interest, we have to define (4) in such a way that the corresponding $<_s$ will be a strict total order on $\mathcal{P}$.

LEMMA 2.1. *Every set of candidate publications $P_s^C(t)$ paired with its strict total order $<_s$ is a totally ordered set.*

PROOF. If $<_s$ is the strict total order on $\mathcal{P}$, then $\mathcal{P}$ paired with it will be a totally ordered set, and every subset of $\mathcal{P}$ will be a totally ordered set too. Therefore, a set $P_s^C(t) \subseteq P^A(t) \subseteq \mathcal{P}$ paired with $<_s$ is a totally ordered set. □

Definition *(More Relevant Set)* Suppose that $s \in \mathcal{S}$ is a subscription and $P \subseteq \mathcal{P}$ is a set of publications, and p' is a publication in the system, where $p' \in P$ or $p' \notin P$. We define a set $P_s^B(p', P) \subseteq P$ of publications from P that are more relevant to a subscription s than p':

$$P_s^B(p', P) \stackrel{\text{def}}{=} \{p \in P : p' <_s p\}. \quad (6)$$

Using a strict order $<_s$ on a set $P_s^C(t)$ we can decide which candidates are better than others, and thereby we will select the best k_s among them for delivery to the subscriber of s. We will refer to these k_s best candidate publications in t as *top-k publications* at t.

Definition *(Top-k Publications)* Suppose that $s \in \mathcal{S}$ is a subscription in the system, and t is a point in time where s is active. We define a set of *top-k publications* $P_s^T(t) \subseteq P_s^C(t)$ for s in t as follows:

$$P_s^T(t) \stackrel{\text{def}}{=} \{p \in P_s^C(t) : |P_s^B(p, P_s^C(t))| < k_s\}. \quad (7)$$

In other words, a publication $p \in P_s^C(t)$ will be an element of $P_s^T(t)$ if there are less then k_s more relevant elements than p in $P_s^C(t)$.

It is important to notice that for different points in time, the corresponding set $P_s^C(t)$ is probably different, because it depends on time. As time passes old publications expire while some new ones are published. Up to this point, we have explained which publications are better than others, and we have defined the top-k publications at some point in time. Now, we will define which publications have to be delivered to a subscriber.

Definition *(Top-k/w Publications)* Suppose that $s \in \mathcal{S}$ is a subscription in the system. We define a set of *top-k/w publications* $P_s^D \subseteq \mathcal{P}$ for s as follows:

$$P_s^D \stackrel{\text{def}}{=} \{p \in \mathcal{P} : p \in P_s^T(t) \wedge t \geq t_s + \delta_s \wedge t \leq \tau_s\}. \quad (8)$$

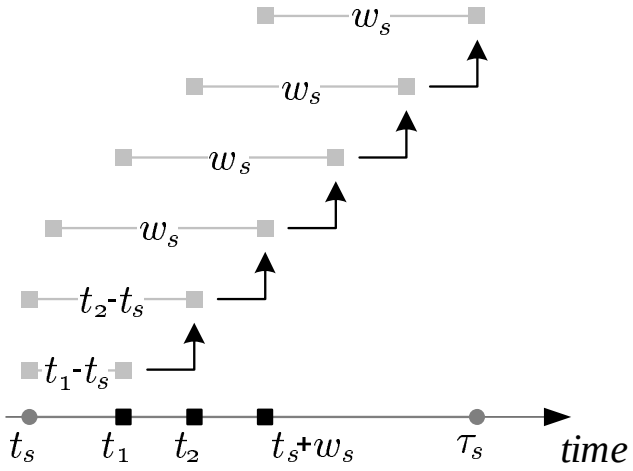

Figure 4: Sliding time window

In other words, a publication $p \in P_s^C(t)$ will be delivered to subscriber of s if it is among top-k publications in at least one moment in time while s is active.

2.1 Sliding Time Window

There is an alternative way of looking at the set $P_s^C(t)$ defined in (3). We can say that for each point in time $t > t_s$ there exists a corresponding time interval $(\max(t_s, t-w_s), t]$ to which we refer as the *time window* $w_s(t)$. The set $P_s^C(t)$ is actually the set of publications published within $w_s(t)$ such that they are still active at t. For $t < t_s + w_s$, the size of the time window is $t - t_s$ and otherwise it is w_s, as shown in Figure 4.

We see that the set $P_s^T(t)$ of subscription s defined in (7) is actually the set of k_s most relevant publications that are published inside a time window $w_s(t)$ such that they are still active at t. We also see that the set P_s^D defined in (8) is actually the set of all top-k publications from every different time window $w_s(t)$ in which s is active.

We can interpret this situation in the following way. For a subscription s there exists a sliding time window $w_s(t)$ that shifts through time. It starts from a point in time $t = t_s$ when s is activated, and continues until $t = \tau_s$ in which s expires. If a publication is among top-k publications in any of these different window positions (i.e. it is a top-k/w publication for some point in time t), it will be delivered to the subscriber of s. It is important to notice that the time window has different sizes at different positions in time, as shown in Figure 4. It is a logical consequence of the fact that the time window of a subscription cannot start before the activation of the subscription, because we cannot guarantee that we will keep the candidate publications of a subscription before we know that it exists.

2.2 Special Cases

In this subsection we discuss how our model behaves for the extreme values of parameters k and w: 1) $k = 0$, 2) $k \to \infty$, 3) $w \to 0$, and 4) $w \to \infty$. The standard case, for which $k \neq \{0, \infty\}$ and $w \neq \{0, \infty\}$, we call the top-k/w case.

Parameter $k = 0$. Suppose that t is a point in time,

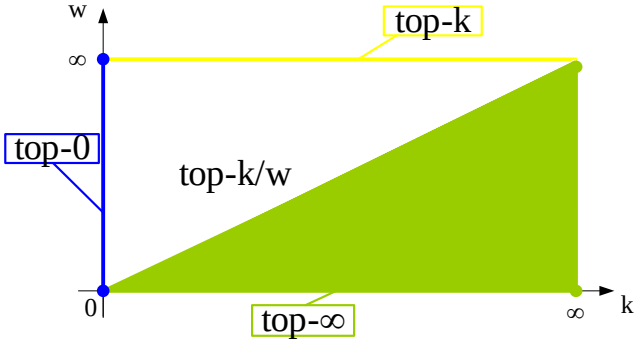

Figure 5: Special cases of the top-k/w model

that $s \in \mathcal{S}$ is an active subscription of a client $c \in \mathcal{C}$ in t for which parameter $k_s = 0$, and that $P_s^C(t)$ is a set of candidate publications of s in t. Then a finite set $P_s^T(t) \subseteq P_s^C(t)$ of top-k/w publications of s in t becomes equal to:

$$[P_s^T(t)]_{k_s=0} = \emptyset. \tag{9}$$

Thus, we conclude that a subscription with parameter $k_s = 0$ is an empty subscription, and regardless of the value of parameter w_s, it will never cause the delivery of any publication to c. We will refer to this spacial case as the top-0 case.

Parameter $k \to \infty$. Suppose that t is a point in time, $s \in \mathcal{S}$ is an active subscription of a client $c \in \mathcal{C}$ in t for which $k_s \to \infty$, and $P_s^C(t)$ is a set of candidate publications of s in t. Then a finite set $P_s^T(t) \subseteq P_s^C(t)$ of top-k/w publications of s in t becomes equal to:

$$[P_s^T(t)]_{k_s \to \infty} = P_s^C(t). \tag{10}$$

In this case, regardless of the value of parameter w_s, all published publications (i.e. all candidate publications) will be delivered to c. We will refer to this spacial case as the top-∞ case.

Parameter $w \to 0$. Suppose that t is a point in time and $s \in \mathcal{S}$ is an active subscription of a client $c \in \mathcal{C}$ in t for which $k_s \nrightarrow 0$. If parameter w_s is sufficiently small, as in this case, every published publication will be the only member of candidate publications of s at the exact moment of its publishing and therefore it will be delivered to c, according to (3). We conclude that this case is also the top-∞ case.

Parameter $w \to \infty$. This is the case where a subscribed client "wants" to receive only k publications among publications published since it has subscribed, such that they are the most related to its subscription. This case is an optimal stopping problem known as the multiple secretary problem, for which the optimal strategy has been proven in [6, 7]. In the top-k/w model we refer to this case as the top-k case.

Low Intensity of Publishing. Suppose that $s \in \mathcal{S}$ is an active subscription of a client $c \in \mathcal{C}$ with some parameters k_s and w_s. In the previous four cases we did not care about the distribution of the publishing of publications. In this case, we will assume that the distribution of publishing is Poisson with an intensity $\lambda < k_s/w_s$. In this case, the expected number of publications that are published during w_s is equal to $\lambda \cdot w_s$, which is smaller than k_s. Therefore all published publications will be delivered to c. This case is also the top-∞ case.

We depict how the combination of parameters k and w influences the top-k/w problem for the five mentioned cases and the standard case with $\lambda > k_s/w_s$ which is the top-k/w case in Figure 5. The standard top-k/w case is analyzed hereafter in the paper.

3. THEORETICAL ANALYSIS OF PUBLICATIONS QUEUE

As previously stated, a practical solution to the top-k/w problem is based on the existence of a sorted set of previously published publications for each subscription, to which we refer as *publications queue*. In this section we deal with the problem of publications queue length. The queue length is very important because shorter queue means faster processing and requires less memory. We can reduce the length of a queue by omitting to store publications which are highly unlikely to ever become top-k/w publications.

As time passes, the newly published publications enter the queue (i.e. they are *born*), while the ones older than w_s are dropped from it (i.e. they *died*). Each publication in the queue belongs to one of the following categories:

1. *Excellent Candidates*—Publications among top-k publications in the queue at the moment of their publication,
2. *Good Candidates*—Publications not among top-k publications in the queue at the moment of their publishing, such that their probability to become a top-k publication at some later point in time before w_s passes is larger than a minimal probability σ, and
3. *Bad Candidates*—Publications for which this probability is less than σ.

We propose a solution to the top-k/w problem which maintains only excellent and good candidates in the queue. Each queue has two parts: 1) the head for keeping excellent candidates, and 2) the tail for keeping good candidates. The length of the head is obviously k, while the length of the tail depends on the minimal probability σ. In this section we answer the following question: *What is the minimal length of the tail of a queue that certainly has all good candidates?*

To answer the previous question we have to make two assumptions. The first assumption is related to the law of publishing and the second to their relevance. First, we assume that new publications are published according to Poisson distribution with an intensity λ. Therefore, the probability that the length of a queue Q is equal to n is

$$p[|Q| = n] = p[B(t) - B(t-w) = n] = e^{-\lambda w} \cdot \frac{(\lambda w)^n}{n!}, \quad (11)$$

where the number of births $B(t)$ is a Poisson random variable with intensity λ. The expected number of births during w is equal to $E[|Q|] = \lambda w = m$. The list of symbols used in this section may be found in Table 1. Second, we assume that relevances of the published publications are independent from their time of publishing, and are mutually independent.

3.1 The Required Length of Tail

We suppose that s is a subscription with parameters k_s and w_s, and that Q_s is its publications queue of the full length (i.e. at some point in time t it contains all candidate publications $P_s^C(t)$). Now, let us suppose that the length of Q_s at some point in time t where a publication p is born is n, and that the starting position of p in Q_s is l. At some point in time $t' > t$, the new position l' of p in Q_s will be the following

$$l' = l - i + j, \quad (12)$$

where i is the number of better publications that have died, and j is the number of better publications that have been born between points in time t and t'. The number of deaths $D(t)$ is also a Poisson random variable with intensity λ, because every publication that is older than w_s dies. The probability that the number of deaths between t and t' is equal to d is

$$p_d(\Delta t) = p[D(t') - D(t) = d] = e^{-\lambda \Delta t} \cdot \frac{[\lambda \Delta t]^d}{d!}, \quad (13)$$

where $\Delta t = t' - t$.

Suppose that d is the number of dead publications. We want to know the probability that among d dead publications, i are better than p. Let us take a look at Figure 6. Every publication in Q_s, except p which is shown as filled square, has an equal probability of dying. Therefore, there are $\binom{n-1}{d}$ different combinations of choosing d among $n-1$ potentially dead publications. One such combination is shown in the figure, where dead publications are shown as crossed squares. Additionally, i of d dead publications are better than p, and thus there are $\binom{l-1}{i}$ different combinations of choosing i among $l-1$ publications better than p. Up to now, we have chosen i of d dead publications, and thus $d-i$ dead publications are still there for choosing among $n-l$ publications worse than p. Thereby, the probability that i of d dead publications were better than p is

Table 1: List of symbols used in this section

Symbol	Meaning
n	length of the queue when p is born
λ	intensity of publishing of publications
m	expected number of births during w_s
l	beginning position of p in the queue
l'	ending position of p in the queue
b	number of births
d	number of deaths
i	number of deaths better than p
j	number of births better than p
r	relative position of p

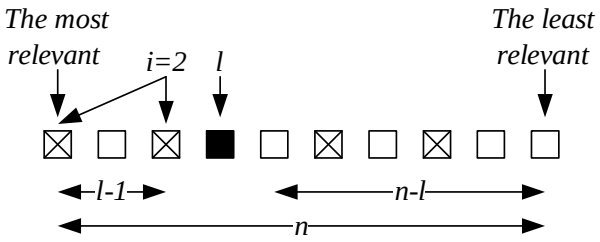

Figure 6: An example queue: The two of the four dead publications are better than p

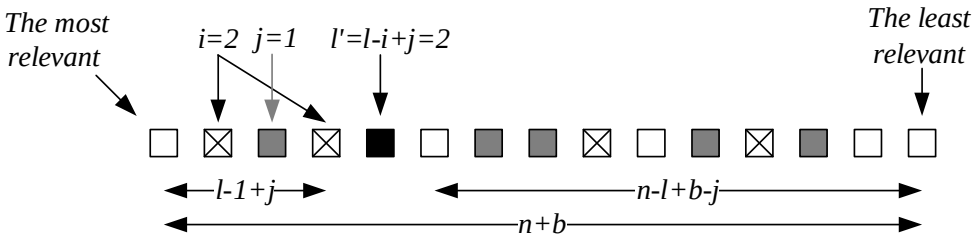

Figure 7: An example queue: One of the five born publications is better than p

$$p_i(l,n,d,i) = \frac{\binom{l-1}{i}\binom{n-l}{d-i}}{\binom{n-1}{d}}, \tag{14}$$

where $l \leq n$, $i \leq l-1$, $i \leq d$ and $d \leq n-1$. Formula (14) is the well known probability mass function of the hypergeometric distribution. Analogously to (13), a probability that the number of births between t and t' is equal to b is:

$$p_b(\Delta t) = p[B(t') - B(t) = b] = e^{-\lambda \Delta t} \cdot \frac{[\lambda \Delta t]^b}{b!}, \tag{15}$$

where $\Delta t = t' - t$.

Suppose that b is the number of born publications. We want to know the probability that among b born publications, j are better than p. This is a conditional probability $p(X|Y)$, where event Y is "the position of p in t is l", and event X is "j of b born publications are better than p in t'". First, let us find $p(X \cap Y)$, i.e. the probability that the position of p in t is l and that among b born publications j are better than p in t'. Let us take a look at Figure 7. Publication p is shown as filled square, the publications older than p are shown as empty or crossed squares, and the publications younger than p are shown as gray squares. From (12), the new position of p in t' is $l' = l - i + j$. There are $l - 1 + j$ publications better than p, and $\binom{l-1+j}{j}$ different combinations of choosing j born publications among them. Analogously, there are $n-l+b-j$ publications worse than p, and $\binom{n-l+b-1}{b-j}$ different combinations of choosing $b-j$ born publications among them. Additionally, there are $\binom{n+b-1}{b}$ different combinations of choosing b born publications among $n+b-1$ equally possible positions, and $n+b$ different ways of picking the position of p. Thereby, the probability $p(X \cap Y)$ is given with the following formula

$$p(X \cap Y) = \frac{\binom{l-1+j}{j}\binom{n-l+b-j}{b-j}}{\binom{n+b-1}{b}(n+b)}. \tag{16}$$

The conditional probability that among b born publications j are better than p is:

$$p_j(l,n,b,j) = p(X|Y) = \frac{p(X \cap Y)}{p(Y)}. \tag{17}$$

The probability of event Y is $1/n$, because there are n possible positions of p in t. Thus, using (16) and (17) we may write the following formula:

$$p_j(l,n,b,j) = \frac{n}{n+b} \cdot \frac{\binom{l-1+j}{j}\binom{n-l+b-j}{b-j}}{\binom{n+b-1}{b}}. \tag{18}$$

where $l \leq n$ and $j \leq b$. Using (14), (12) and (17) we may

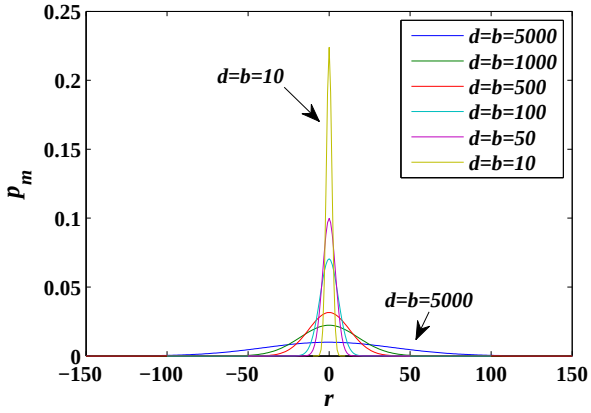

Figure 8: Probabilities of relative positions for $l = 2000$ and $n = 10000$

write the probability $p_m(l', l, n, d, b)$ that p is at a new position l' after b births and d deaths as:

$$p_m(l', l, n, d, b) = \begin{cases} \sum_{k=|r|}^{\min(l-1,d)} p_i(l,n,d,k) \cdot p_j(l,n,b,k-|r|) & \text{if } l' \leq l \\ \sum_{k=|r|}^{\min(l'-1,b)} p_i(l,n,d,k-|r|) \cdot p_j(l,n,b,k) & \text{else} \end{cases} \tag{19}$$

where $r = l' - l$ is a relative position of p in t'.

In Figure 8 we can see the probability of each possible r of p as a function of the same number of births and deaths, for $l = 2000$ and $n = 10000$. It is important to notice that its relative new position is limited by the number of births and deaths. For example, for 10 births and deaths, the extreme relative positions are: 1) $r = -10$ for $j = 0$ and $i = 10$ (i.e. 10 births of worse publications and 10 deaths of better publications), and 2) $r = 10$ for $j = 10$ and $i = 0$ (i.e. 10 births of better publications and 10 deaths of worse publications). As we can see from the graph, for a larger number of births and deaths, the extreme relative positions are larger too, but their probability drops fast. These extreme positions will be maximal for the maximal number of births and deaths, and this always happens just a moment before p dies (i.e. $t' = t + w_s$). In that moment, there are no more publications older than p in Q_s, and therefore its position depends only on the born publications, and we may

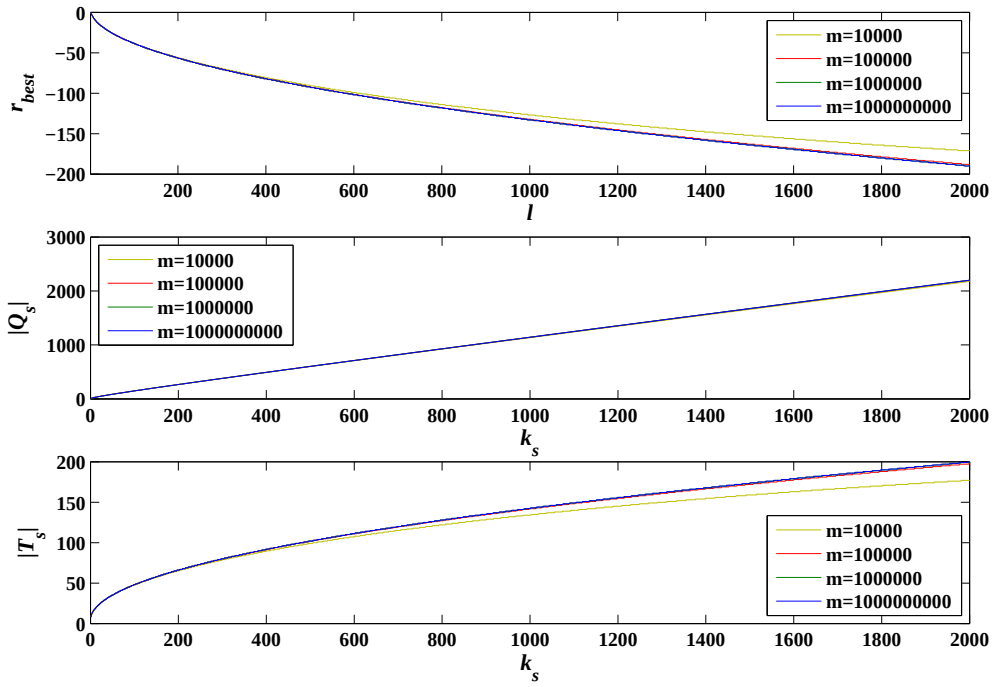

Figure 9: Results of calculations

write (19) in the following way:

$$p_e(l', l, n, b) = p_m(l', l, n, n-1, b) = p_j(l, n, b, j), \quad (20)$$

where $j = l' - l + i = l' - l + (l-1) = l' - 1$. From (15) we know that the number of births between t' and t is a Poisson random variable with the expected value $E[B(t') - B(t)] = \lambda w = m$. If we sum (20) over all possible values of b multiplied by their probabilities, we will get the probability $p_f(l', l, n, m)$ of every new position of p in t' using the following formula:

$$\begin{aligned} p_f(l', l, n, m) &= \sum_b p_b(w) \cdot p_j(l, n, b, j) = \\ &\sum_b e^{-\lambda w} \cdot \frac{(\lambda w)^b}{b!} \cdot p_j(l, n, b, j) = \\ &\sum_b e^{-m} \cdot \frac{(m)^b}{b!} \cdot p_j(l, n, b, j), \end{aligned} \quad (21)$$

where $j = l' - 1$. It is possible that instead of summing up over all possible values of b multiplied by their probabilities, we just put in (20) the expected value m of the number of births between t' and t:

$$\begin{aligned} p_a(l', l, n, m) &= p_j(l, n, m, j) = \\ &\frac{n}{n+m} \cdot \frac{\binom{l-1+j}{j}\binom{n-l+m-j}{m-j}}{\binom{n+m-1}{m}}. \end{aligned} \quad (22)$$

where $j = l' - 1$. Our simulations show that (21) can be very well approximated with (22), thus $p_f(l', l, n, m) \approx p_a(l', l, n, m)$. For a publication p that was at a position l in t, we may get a probability of each new position l' in $t' = t + w$ by fixing n and m in (21).

At this point, we have to make an important remark. Let us recall what is n in (22). It is the length of Q_s at some point in time t where our publication of interest p is born. Thereby, to calculate probabilities of new positions of different publications, we have to know the length of Q_s at a point in time when each publication is born. This is very impractical, so instead of the exact value we will take the expected value of this length, which is m. Therefore, equation (22) becomes the following formula:

$$\begin{aligned} p_u(l', l, m) &= p_a(l', l, m, m) = \\ &\frac{m}{2m} \cdot \frac{\binom{l-1+j}{j}\binom{m-l+m-j}{m-j}}{\binom{m+m-1}{m}} = \\ &\frac{1}{2} \cdot \frac{\binom{m}{j}\binom{m-1}{l-1}}{\binom{2m-1}{l-1+j}}, \end{aligned} \quad (23)$$

where $j = l' - 1$. In Section 1.2 we mentioned that a window w can be either time-based or number-based. For the number-based windows (23) is exact, because the length of Q_s is always m.

Now, let us return to time-based windows, we would like to know which new positions have probabilities less than σ to become a top-k publication at some later point in time before w_s passes. To find that out, we have to sum the probabilities of becoming a top-k publication, i.e. probabilities of positions $l' = 1 \ldots k$. For each position l this sum is given

with the following formula:

$$p_\Sigma(l,m) = \sum_{l'=1}^{k} p_u(l',l,m). \quad (24)$$

The first position that has more than the minimal probability is thus:

$$l_{best} = l'' \in \mathbb{N} : p_\Sigma(l'',m) \geq \sigma \wedge p_\Sigma(l''+1,m) < \sigma. \quad (25)$$

We have run four different calculations for the values l of s from 1 to 2000, and the following values of m: 10000, 100000, 1000000 and 1000000000. The minimal probability σ was set to 0.001. The results are shown in Figure 9, where $|Q_s|$ is the length of Q_s, and $|T_s|$ is the length of the tail of Q_s. From this figure, we can conclude the following two very important facts:

- The lengths of Q_s and T_s *almost do not depend* on m, and
- The lengths of Q_s and T_s *grow sub-linearly* with k_s.

Because of the first fact, while identifying the required length of Q_s, we can take the worst value of m (i.e. 1 billion), and it will just slightly increase the length of Q_s compared to the length for smaller values of m. From Figure 9, it is visible that this slight increase occurs for larger values of k_s, which is good, because subscribers will probably choose smaller values for k_s.

4. EXPERIMENTAL RESULTS

For a systematic analysis of our approach we used the same representation (of publications and subscriptions) and datasets as Böhm at al. have used in [4]. We have chosen to compare our approach with theirs because they did an extensive analysis of the problem very similar to ours. In this section, we have three different goals. First, to compare the number of delivered publications of our probabilistic approach with the top-k/w model. Second, to compare simulation runtimes for different values of k and different data dimensionality. Third, to compare runtime and number of delivered publications for different values of parameters k and w.

The following setup was used in experiments: Publications and subscriptions are points in a multidimensional attribute space. Relevances are Euclidean distances between subscriptions and publications. We generated synthetic datasets of various dimensionality. In particular, we used clustered Gaussian and uniformly distributed data. The clustered datasets contained ten clusters with standard deviation 0.1, which were randomly distributed on the attribute space. Additionally, we used LBL-TCP 3 dataset, which is a real netflow dataset available at Internet Traffic Archive[4]. For all experiments, we used 1 million of publications. Additionally, we used 500 subscriptions and assumed that each subscription has the same number-based[5] sliding window of size 20000 when not otherwise specified. We further assumed that at each time stamp a new publication is published. Experiments were run in Java on a PC with 2.8 GHz Pentium Processor and 1 GB of main memory. Similarly to [4], we represented publications queues in the main memory as two double-linked lists. In the first list, the publications were sorted by their relevance, while in the second list they were sorted by their time of publishing.

[4]http://ita.ee.lbl.gov

[5]See the first part of Section 1.2 for the explanation.

Table 2: Number of delivered publications for different datasets

	Uniform		Clustered Gaussian		Netflow	
k	a	b	a	b	a	b
1	102	102	104	104	905	1019
2	194	194	198	198	1139	1307
5	418	418	410	410	1662	1685
10	760	760	784	784	2299	2200
20	1382	1382	1393	1393	3263	3264
50	3196	3196	3217	3216	6152	5910
100	6068	6068	6114	6114	9774	9147
200	11728	11728	11749	11749	15964	15326
500	28096	28096	28018	28018	40488	39173
1000	54631	54631	54739	54739	71447	71614
2000	106738	106738	106638	106638	126169	127133

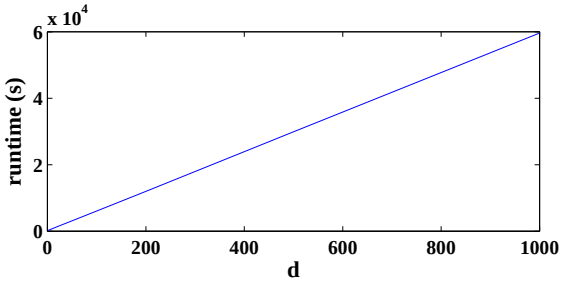

Figure 10: Runtime for different dimensionality

In Table 2 we can see the number of delivered publications for a **randomly chosen subscription** and different values of parameter k. For each dataset, column a is number of delivered publications for tail of endless length (as it is defined in the model), and column b is the same number for tail with the required length determined in Section 3. For the synthetic datasets these numbers are identical. Actually, the only difference is for clustered Gaussian data and $k = 50$. Therefore, we conclude that our approach behaves as expected with less than 0.1% of non-delivered top-k/w publications. For the netflow dataset these numbers are different, which is the consequence of repeated identical publications (i.e. source port, destination port and packet size) and high dependence between publishing times and relevances in the dataset. As previously stated, our model is based on assumption that each publication is unique, while our solution is based on assumption that relevances of publications to a subscription are independent of their time of publishing. Thus, we conclude that it behaves reasonably well for this problematic kind of data. It is important to notice that because of the identical publications, the number of delivered publications is few times larger in the case of the netflow dataset compared to the synthetic datasets.

In Figure 10 we can see the runtime of simulation for the uniform dataset and different number of attribute space dimensions. Each of 500 subscriptions had the same value of parameter $k = 1$. We conclude that the runtime grows linearly with the number of dimensions. This was expected because the complexity of calculating of Euclidean distance

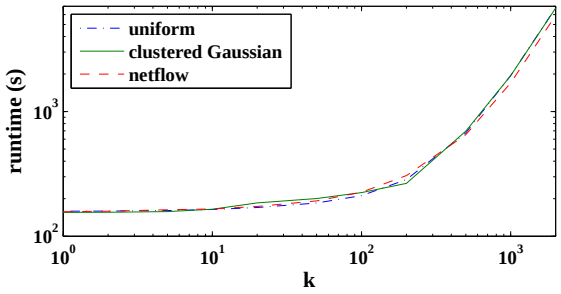

Figure 11: Runtime for different datasets

in a multidimensional attribute space grows linearly with the number of its dimensions.

In Figure 11 we can see the runtime of simulation in **2-dimensional attribute space** for different datasets and different values of parameter k. We conclude that there is no significant difference in the runtime for different datasets. Now, let us analyze the runtime for different values of parameter k. We conclude that runtime stays between 200 and 300 seconds for the values smaller than $k = 100$. For the values $k > 100$, runtime grows almost linearly, and this is the consequence of using sorted linked lists as publications queues, where inserting of a new element takes $O(n)$ [12]. Therefore, we conclude that the runtime may be additionally improved for larger values of k by using balanced binary trees (with $O(log_2 n)$) instead of double-linked lists.

Now, it is important to remark that in [4], the authors use query indexing algorithm and delaying of newly arriving objects that we do not. We simply check all the subscriptions upon a new publishing event. According to simulation in [4], the runtime of simple approach without indexing and delaying is for $k = 1$ equal to $11.58s \cdot 63 = 729.54s$. As we can see from the beginning of the graph in Figure 11, the runtime of our approach is for $k = 1$ equal to $153.50s$. Therefore, we conclude that our approach is for $k = 1$ more than $4,5$ times faster than the simple approach. Moreover, presented results have been performed without indexing and our algorithm is for values of $k > 10$ faster than their approach with indexing and delaying. Furthermore, we are working on a subscription indexing algorithm, which will according to [4] drastically improve performances of our approach.

In Figure 12 we can see the number of delivered publications for a **randomly chosen subscription** and different values of parameters k and w. We used number-based windows and the uniform dataset in 2-dimensional attribute space. As we know from Section 2.2, if k is more or equal than the number of published publications in window w, all of them will be delivered to the subscriber. For example, there are 1 million delivered publications for $k = w = 100$. It is very important to notice that the numbers of delivered publications will be very similar if we increase (or decrease) both k and w for the same orders of magnitude. For example, the number of delivered publications is almost identical for pairs $(k, w) = (10^1, 10^3)$ and $(10^2, 10^4)$.

Now, it would be interesting to see which of possible (k, w) pairs requires less processing for approximately the same amount of delivered publications. In Figure 13 we can see runtimes for different (k, w) pairs. The upper-right part represents runtimes of the top-∞[6]case. For these pairs, all

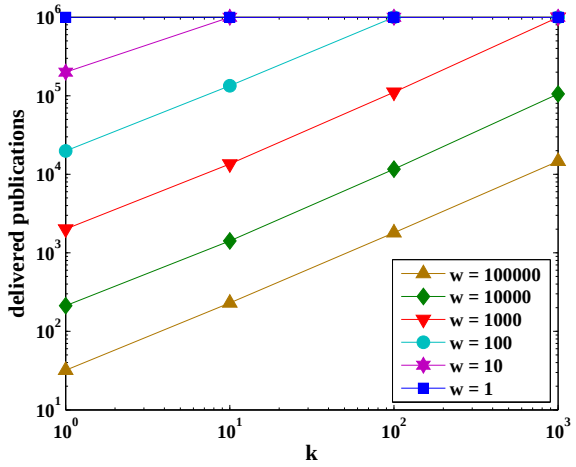

Figure 12: Number of delivered publications for different values of parameters k and w

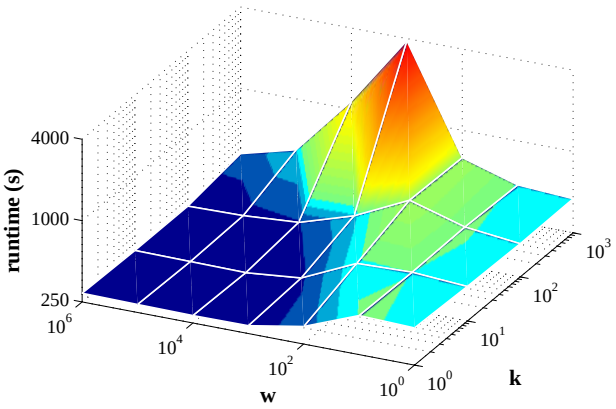

Figure 13: Runtime for different values of parameters k and w

the published publications will be delivered to subscriber. To reduce unnecessary processing, this case should be recognized at the moment of subscription activation. As we see, the peak is for the biggest simulated value of $k = w$ (i.e. 10^3). This is expectable, because in this case, we have the longest of the publications queues that contain every active publication. The runtime is large because we have to put each newly published publication to its right position it the queue, although it will certainly be delivered to the subscriber. The lower-left part represent runtimes of the standard top-k/w[6] case. In this case, our advice is to choose one of the pairs in the middle, which have better runtimes than outer pairs. For example, it is better to choose pairs $(10^2, 10^4)$ or $(10, 10^3)$ than pairs $(1, 10^2)$ or $(10^3, 10^5)$.

5. RELATED WORK

There are many different ways for the representation (i.e. content and structure) of publications and subscriptions in CBPS systems. The common representation of publications

[6]See Section 2.2 for the explanation.

is as attribute-value pairs in a multi-attribute space, and subscriptions as Boolean functions of simple constraints on the attributes of the same multi-attribute space [9, 15, 32, 30, 29]. This representation is completely structured. Another very popular representation is XML representation [11, 10, 36], which is semi-structured. The matching model for all of this representations is essentially Boolean in nature, and suffers from the drawbacks mentioned in Section 1.1.

The idea that a publication has a degree of relevance to a subscription, instead of simply being matching or non-matching, is not a new one [22, 24, 23, 8, 31, 20]. Liu and Jacobsen [22, 24, 23] introduced the degree of relevance in publish/subscribe systems. The main part of their research is focused on expressing of uncertainties in publications and subscriptions, and the approximate matching of such publications and subscriptions. For each match, they compute the two measures: possibility and necessity of match. Their approximate matching model uses a number of parameters to control the tolerance of a match on very fine-granular basis. This is very different from our model. In our model these parameters are not defined in each subscription, because we use the relevance of other publications to control the number of matches instead, see Figure 2. Caporuscio and Inverardi [8] define confidence of matching, which is a measure similar to our degree of relevance. They use this measure for modeling of uncertainties. The main goal of their work is to achieve event (i.e. publication) correlation through the use of complex filters. Picco at al. [31] deal with the uncertainties in the tuple space model. Lekova et al. [20] use fuzzy reasoning for filtering and matching of numerical values in publications to imprecise data in subscriptions.

In the above systems, the degree of relevance of publications to subscriptions is the direct consequence of their fuzzy representations. On the contrary, in our model it is the starting point. Since our model does not depend on a representation of publications and subscriptions, we do not care about the domain of interest and the corresponding representation, as long as the representation supports calculation of the relevance. Without a capability of calculating the relevance, our model is not applicable for the chosen representation in the corresponding domain of interest. There are many different techniques for calculating relevance [2] for different representations of queried objects. Some of these techniques are intended for structured representations [5, 16], and some for unstructured [35, 34]. In some of these techniques relevance depends only on the representations, and in others it also includes some additional information. Our model is general enough that it can be applied in all these cases.

The most similar problem to top-k/w publish/subscribe is processing of continuous top-k queries on data streams [17, 19, 27, 4, 13]. A survey of management and processing of data streams can be found in [1]. Böhm at al. [4] develop a criterion to decide upon the arrival of a new object if it may become the k-NN (k-nearest neighbor) or not. It is based on the idea that a new object arriving from the stream can often exclude many other objects which cannot become k-NN until the new object expires. This is actually just one of the possible pruning strategies, which minimizes the number of stored objects. They use *query skyline*—a skyline-based object buffer associated with each query to keep potential k-NN. See [21] for the explanation of skyline. This buffer is very similar to publications queue in our approach. Opposite to our approach which is probabilistic, their approach is purely deterministic and gives an exact answer. They also propose delaying of new objects, and a very fast query indexing algorithm. They did an extensive experimental evaluation of their approach on high throughput data streams, comparing it with a simple approach without the indexing and the delaying. As this is the most similar problem to ours, and this simple approach is actually the reference point to compare with, we compared the two approaches in Section 4. It is important to notice that representing publications and subscriptions as points is a multi-dimensional space is just one of their possible representations. Gao and Wang [17] propose discovering of patterns to predict content of new objects. Their approach is not applicable if the content of an object is independent of its time of arrival, which is our assumption. Koudas et al. [19] propose approximate k-NN answers with guaranteed error/performance bounds. Das et al. [13] introduce a novel geometric representation of top-k query answering problem. They also propose an index for such objects, object pruning methods for the index, and algorithms for its updating and querying. Their approach is also deterministic as [4]. Mouratidis et al. [27] employ a regular grid to index objects in present window. Their methods relies on either precomputing or recomputing of results of top-k queries and is therefore quite different from our approach.

6. CONCLUSION AND FUTURE WORK

In this paper we presented the drawbacks of the current CBPS systems and proposed the top-k/w publish/subscribe model—a new CBPS model that is based on sliding window computation model, and which does not suffer from the listed drawbacks. Our model supports different representations of publications and subscriptions as long as they allow calculation of relevance. We presented the practical implementation of our model and developed a probabilistic criterion to decide if a publication is (upon its arrival) worth of keeping in the system or not. We showed that this criterion allows us to process very large number of publications in a short period of time.

The most important consequence of our approach is that we finally can have a publish/subscribe system with completely unstructured representations of publications and subscriptions. Therefore, this very popular data representation is no more exclusively intended to be used in information retrieval systems. The second very important consequence is that for the first time we present a solution for too general and over-specified subscriptions.

We are working on a fast matching algorithm based on subscription indexing, which supports subscription coverings. We are thinking of incorporating some kind of penalties for publications that are waiting in a publications queue. The more publications wait they will be less relevant, because in some applications scenarios they actually loose their value to subscribers with time. We are currently working on a distributed system based on our model, which uses rendezvous-based routing [3].

7. REFERENCES

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS*, pages 1–16, New York, NY, USA, 2002. ACM.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[3] R. Baldoni and A. Virgillito. Distributed Event Routing in Publish/Subscribe Communication Systems: a Survey (revised version). Technical report, Dipartimento di Informatica e Sistemistica "A.Ruberti", Universitá di Roma la Sapienza, 2006.

[4] C. Böhm, B. C. Ooi, C. Plant, and Y. Yan. Efficiently processing continuous k-nn queries on data streams. In *ICDE*, pages 156–165. IEEE, 2007.

[5] N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. Database Syst.*, 27(2):153–187, 2002.

[6] F. Bruss. Sum the odds to one and stop. *Annals of Probability*, 28(3):1384–1391, 2000.

[7] F. Bruss and D. Paindaveine. Selecting a sequence of last successes in independent trials. *Journal of Applied Probability*, 37:389–399, 2000.

[8] M. Caporuscio and P. Inverardi. Uncertain event-based model for egocentric context sensing. In *SEM*, pages 25–32, New York, NY, USA, 2005. ACM.

[9] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *PODC*, pages 219–227, New York, NY, USA, 2000. ACM Press.

[10] R. Chand and P. Felber. Xnet: A reliable content-based publish/subscribe system. In *SRDS*, pages 264–273, Washington, DC, USA, 2004. IEEE Computer Society.

[11] R. Chand and P. Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *Euro-Par*, pages 1194–1204, 2005.

[12] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[13] G. Das, D. Gunopulos, N. Koudas, and N. Sarkas. Ad-hoc top-k query answering for data streams. In *VLDB*, pages 183–194. VLDB Endowment, 2007.

[14] M. Datar and R. Motwani. *The Sliding-Window Computation Model and Results*, chapter 8, pages 149–167. Advances in Database Systems. Springer-Verlag New York, 2006.

[15] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *SIGMOD Rec.*, 30(2):115–126, 2001.

[16] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.

[17] L. Gao and X. S. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *SIGMOD*, pages 370–381, New York, NY, USA, 2002. ACM.

[18] S. Kale, E. Hazan, F. Cao, and J. P. Singh. Analysis and algorithms for content-based event matching. In *ICDCSW*, pages 363–369, Washington, DC, USA, 2005. IEEE Computer Society.

[19] N. Koudas, B. C. Ooi, K.-L. Tan, and R. Zhang. Approximate nn queries on streams with guaranteed error/performance bounds. In *VLDB*, pages 804–815. VLDB Endowment, 2004.

[20] A. Lekova, K. Skjelsvik, T. Plagemann, and V. Goebel. Fuzzy logic-based event notification in sparse manets. In *AINAW*, pages 296–301, Washington, DC, USA, 2007. IEEE Computer Society.

[21] X. Lin, Y. Yuan, W. Wang, and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In *ICDE*, pages 502–513, Washington, DC, USA, 2005. IEEE Computer Society.

[22] H. Liu and H.-A. Jacobsen. A-topss - a publish/subscribe system supporting approximate matching. In *VLDB*, pages 1107–1110, 2002.

[23] H. Liu and H.-A. Jacobsen. A-topss – a publish/subscribe system supporting imperfect information processing. In *VLDB*, pages 281–1284, August 2004.

[24] H. Liu and H.-A. Jacobsen. Modeling uncertainties in publish/subscribe systems. In *ICDE*, page 510, Washington, DC, USA, 2004. IEEE Computer Society.

[25] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.

[26] S. Michel, P. Triantafillou, and G. Weikum. Klee: a framework for distributed top-k query algorithms. In *VLDB*, pages 637–648. VLDB Endowment, 2005.

[27] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *SIGMOD*, pages 635–646, New York, NY, USA, 2006. ACM.

[28] G. Mühl, L. Fiege, and A. P. Buchmann. Filter similarities in content-based publish/subscribe systems. In *ARCS*, pages 224–240, London, UK, 2002. Springer-Verlag.

[29] G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[30] G. Mühl, A. Ulbrich, K. Herrmann, and T. Weis. Disseminating information to mobile clients using publish-subscribe. *IEEE Internet Computing*, 8(3):46–53, 2004.

[31] G. P. Picco, D. Balzarotti, and P. Costa. Lights: a lightweight, customizable tuple space supporting context-aware applications. In *SAC*, pages 413–419, New York, NY, USA, 2005. ACM.

[32] P. R. Pietzuch and J. M. Bacon. Hermes: A distributed event-based middleware architecture. *ICDCSW*, 00:611, 2002.

[33] C. Raiciu, D. S. Rosenblum, and M. Handley. Revisiting content-based publish/subscribe. In *ICDCSW*, page 19, Washington, DC, USA, 2006. IEEE Computer Society.

[34] S. E. Robertson and K. S. Jones. *Relevance weighting of search terms*. Taylor Graham Publishing, London, UK, UK, 1988.

[35] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *J. ACM*, 15(1):8–36, 1968.

[36] T. Sivaharan, G. S. Blair, and G. Coulson. Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing. In *OTM Conferences (1)*, pages 732–749, 2005.