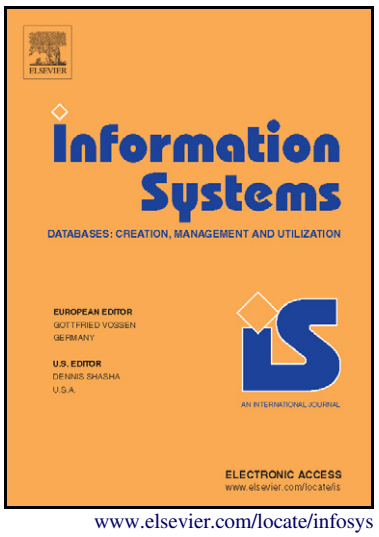


www.elsevier.com/locate/infosys

Top-k/w publish/subscribe: A publish/subscribe model for continuous top-k processing over data streams

Krešimir Pripužić, Ivana Podnar Žarko, Karl Aberer

PII: S0306-4379(12)00049-X
DOI: doi:10.1016/j.is.2012.03.003
Reference: IS781

To appear in: *Information Systems*

Cite this article as: Krešimir Pripužić, Ivana Podnar Žarko and Karl Aberer, Top-k/w publish/subscribe: A publish/subscribe model for continuous top-k processing over data streams, *Information Systems,* doi:10.1016/j.is.2012.03.003

Top-k/w publish/subscribe: A publish/subscribe model for continuous top-k processing over data streams

Krešimir Pripužić[a], Ivana Podnar Žarko[a], Karl Aberer[b]

[a] *Faculty of Electrical Engineering and Computing, University of Zagreb, HR-10000 Zagreb, Croatia*
[b] *School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne, CH-1007 Lausanne, Switzerland*

Abstract

Continuous processing of top-k queries over data streams is a promising technique for alleviating the information overload problem as it distinguishes relevant from irrelevant data stream objects with respect to a given scoring function over time. Thus it enables filtering of irrelevant data objects and delivery of top-k objects relevant to user interests in real-time. We propose a solution for distributed continuous top-k processing based on the publish/subscribe communication paradigm—top-k publish/subscribe over sliding windows (*top-k/w publish/subscribe*). It identifies k best-ranked objects with respect to a given scoring function over a sliding window of size w, and extends the publish/subscribe communication paradigm by continuous top-k processing algorithms coming from the field of data stream processing.

In this paper, we introduce, analyze and evaluate the essential building blocks of distributed top-k/w publish/subscribe systems: First, we present a formal top-k/w publish/subscribe model and compare it to the prevailing Boolean publish/subscribe model. Next, we outline the top-k/w processing tasks performed by publish/subscribe nodes and investigate the properties of supported scoring functions. Furthermore, we explore potential routing strategies for distributed top-k/w publish/subscribe systems. Finally, we experimentally evaluate model properties and provide a comparative study investigating traffic requirements of potential routing strategies.

Keywords: event-based systems, data stream, sliding-window queries.

1. Introduction

The problem of information overload is apparent in our daily lives: When checking RSS feeds from major media sources, browsing through blog posts and status updates on social sites, or following tweets to find breaking news, it is difficult to distinguish information relevant to personal interest in real-time. Moreover, emerging data stream applications spanning over largely-distributed data sources, such as the Sensor Web [1] or large-scale opportunistic sensing [2], continuously generate huge data volumes and require continuous processing and data filtering followed by timely delivery of relevant information, e.g. alerts, to enable human intervention.

We addresses continuous processing of top-k queries over distributed data streams which filters out irrelevant data stream objects, and delivers only top-k objects relevant to current interests in real-time. Thus it offers the means to alleviate the information overload problem facing numerous data stream processing applications where recent objects are more important than older ones. Our solution—top-k publish/subscribe over sliding windows (*top-k/w publish/subscribe*)—identifies k best-ranked objects with respect to a given scoring function over a sliding window of size w, and is based on the publish/subscribe communication paradigm augmented by algorithms coming from the field of data stream processing.

Top-k/w publish/subscribe can be applied in a number of real-world applications: 1) Consider an application which aggregates product advertisements from a number of auction sites. A user may specify properties of an "ideal" product he/she is interested in, while the application informs the user in real-time about newly appearing advertisements that are most similar to his/her "ideal" product. Furthermore, the user can specify the number of advertisements he/she is willing to receive per day. 2) A personal information filtering engine continuously checks for updates from various RSS feeds and blogs, and delivers k most relevant updates to users during a course of a day and at the time when those updates are published. 3) Another possible application area is real-time environmental monitoring. With top-k/w publish/subscribe environmental scientists can identify and monitor up to k sites with the largest pollution readings over the course of a single day, or identify a predefined number of sensors closest to a particular location measuring the largest pollution levels over time (e.g. top-10 readings per hour). All these example applications require continuous processing of top-k queries over largely-distributed data sources generating data objects possibly with high publication rates.

In this paper we present top-k/w publish/subscribe, a novel publish/subscribe model for continuous top-k processing over

Email addresses: kresimir.pripuzic@fer.hr (Krešimir Pripužić), ivana.podnar@fer.hr (Ivana Podnar Žarko), karl.aberer@epfl.ch (Karl Aberer)

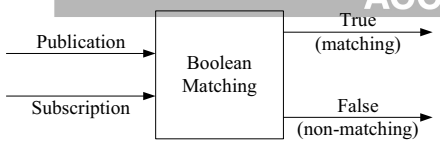


Figure 1: Matching in Boolean publish/subscribe systems

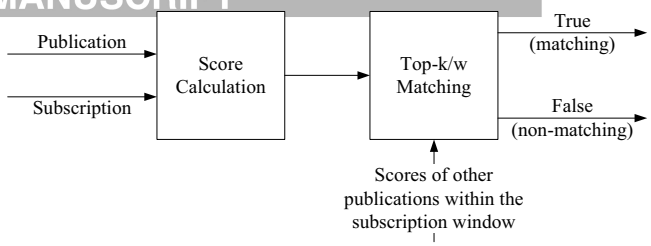


Figure 2: Matching in top-k/w publish/subscribe systems

distributed data streams which introduces a stateful matching model compared to the stateless Boolean model which is widespread in literature. *Publish/subscribe systems* process continuous queries, i.e. *subscriptions*, over distributed sources producing data objects, i.e. *publications*, with the goal of timely notification delivery across distributed data destinations [3]. They are tuned to filter large amounts of published information in real-time and deliver matching publications to interested subscribers due to efficient matching and routing algorithms, but at the expense of supporting rather simple stateless queries. *Data stream processing systems* enable efficient processing of continuous queries, but in contrast to the former systems, they are tightly coupled, platform dependent, difficult to deploy and maintain, and less scalable to the number of users [4]. Our model reuses the best properties of both systems to identify a limited number of most relevant publications over time and enable effective filtering of distributed data streams while providing the means to control the frequency of delivered publications.

1.1. Motivation

In current publish/subscribe systems, subscription is a stateless Boolean function [5]: A decision whether to deliver a publication to a subscriber is made based on the result of the matching process comparing a publication to subscriber's subscription, as shown in Figure 1. The matching process depends exclusively on publication and subscription content, and does not take into account any additional information present in the system. This approach has the following drawbacks:

- a subscriber may be either overloaded with publications or receive too few publications over time,
- it is impossible to compare different matching publications with respect to a subscription as ranking functions are not defined, and
- partial matching between subscriptions and publications is not supported.

As publication content is generally unknown in advance, it is impossible to predict the number of future publications matching a subscription. If a subscription is *too general*, a subscriber may receive too many publications over time. On the contrary, in case of an *overspecified* subscription, the subscriber may receive too few publications or none in the worst case. Thus, a subscriber has to specify an "perfect" subscription to receive an optimal number of matching publications. It is a sort of guessing, where even a slight change in subscription may result in a drastically different number of matching publications. In general, a user perceives 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 too few publications might be recognized as non-working. The number of received publications is crucial for the acceptance of an actual system by users even more if, for example, subscribers pay for each delivered publication matching subscriber information interest.

We argue that the unpredictable number of delivered publications to subscribers over time is one of the reasons for the slow adoption of large-scale publish/subscribe solutions. Either too many or too few publications may cause user dissatisfaction with a provided service, for example, in applications such as personal information filtering, 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.

1.2. Top-k/w Matching Model in Brief

The *top-k/w publish/subscribe model* enables a subscriber to control the number of publications it receives per subscription within a predefined time period. Each subscription defines 1) a time-independent scoring function and 2) the parameters $k \in \mathbb{N}$ and $w \in \mathbb{R}^+$. Unlike the Boolean publish/subscribe model, it is time-dependent because at each point in time t, the parameter k limits the number of matching publications restricting it to top-k publications that are published between points in time $t - w$ and t. This interval is called the *time-based window* of size w and it is constantly *sliding* in time. Please note that the size of subscription window may also be defined as the number of most recent publications (*count-based window*) [6]. With no loss of generality in this paper we assume time-based windows.

The quantity of received publications in the top-k/w publish/subscribe model is independent of the publishing frequency, and depends on parameters k and w. This in practice means that the matching process comparing a newly published publication to a subscription depends both on publication score and scores of previous publications calculated using the same scoring function. In other words, each publication is competing with other publications from the sliding window for a position among the top-k publications as depicted in Figure 2. Obviously, the quality of received publications depends on calculated scores, and is statistically proportional to the number of published publications.

Efficient processing of top-k/w subscriptions is challenging because, even though a publication is not a top-k publication at the time when it is published, it might become one in future,

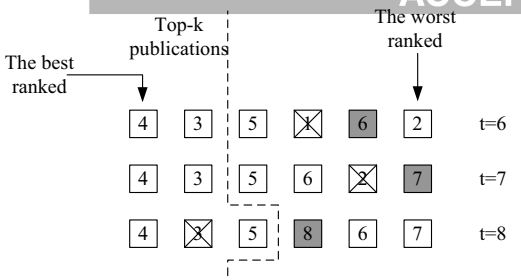


Figure 3: Processing publications for a top-k/w subscription.

and therefore a set of *candidate publications* within the sliding window has to be stored[1] in memory. This is shown by the following example: For a top-k/w subscription with parameters $k = 2$ and $w = 5s$, Figure 3 shows the processing of arriving publications. We depict discrete points in time $t = 6, 7, 8s$ when publications marked with gray boxes, i.e. 6, 7, and 8, arrive in the system. Note that all publications are denoted by boxes and their time of appearance, while we rank them from left to right. For example at $t = 6$, the publication published at $t = 4$ has the best rank, while the publication published at $t = 2$ has the worst rank. When the window slides to $t = 6$, the publication published at $t = 1$ denoted by a checked box is dropped from the window, while a new one marked with a gray box appears in the window. From the presented sequence of events, we can see the publication published at $t = 5$ becomes a top-2 publication at $t = 8$ although it was not among top-2 publications when it was published. Thus publication 5 should be stored in memory at the time it appeared in the window as a candidate publication.

The example shows that a publication within the window of a top-k/w subscription becomes a top-k/w publication when there are less than k higher ranked publications within the window. This can happen either when a new publication enters the system when there are less than k higher ranked publications within the subscription window, or at a later point in time when one of top-k publications is dropped from the subscription window, and a candidate publication has the highest rank among all other candidate publications. Therefore, the set of candidate publications needs to be maintained in main memory for top-k/w processing.

1.3. Contribution

Continuous top-k queries have already been identified as one of the most important types of continuous queries [7], and the data stream community already offers efficient algorithms for continuous top-k processing which maintain the minimal set of candidate publication on a single processing node [8, 9]. However, existing solutions require centralized processing which generates huge network traffic due to transport of data streams to the processing site, and therefore distributed processing and filtering of produced data close to data sources is required for scalable continuous top-k processing.

[1]Note that publications are seen only when entering the system unless they are explicitly stored in memory.

The goal of this paper is to present a distributed solution for top-k/w processing which reuses existing algorithms for continuous top-k/w processing at the nodes forming a distributed overlay network, and extends the principles of publish/subscribe for overlay organization and routing algorithms. Our initial idea to design a stateful matching model for publish/subscribe systems which ranks publications with respect to a subscription is presented in [10], where we also introduce an algorithm for probabilistic computation of top-k/w queries. This idea was developed in parallel with the following two papers [11, 12] that also introduce ranking functions into publish/subscribe systems, but do not use sliding windows to limit the temporal scope of subscription processing. Compared to [10], this paper introduces an extended top-k/w model for distributed publish/subscribe systems, discusses the scoring functions supported by the model, and identifies the routing algorithms that are adequate for top-k/w publish/subscribe. This paper provides a general view on top-k/w publish/subscribe, while a specific solution for processing continuous k-NN (nearest neighbor) queries in a distributed setting is presented in [13].

In particular, the main paper contributions may be summarized as follows:

- The paper presents a formal model of top-k/w publish/subscribe system and compare it to the prevailing model of Boolean publish/subscribe system. We show that the top-k/w publish/subscribe model can be reduced to the Boolean model. The initial version of the top-k/w publish/subscribe model is introduced in [10].
- We identify and analyze typical scoring functions supported by our model.
- We provide an extensive analysis of the existing routing strategies found in publish/subscribe systems, and adapt them to the top-k/w publish/subscribe model. The analysis is used to identify prospective routing strategies for the novel publish/subscribe model.
- An experimental evaluation is used to investigate performance properties of the top-k/w publish/subscribe model for different data sets (one real and two generated data sets). Furthermore, an analytical evaluation of the routing strategies under a hypothetical network scenario is given to quantify the messaging overhead of the introduced top-k/w publish/subscribe model.

The paper is organized as follows. Section 2 presents the Boolean and top-k/w publish/subscribe models and discusses model-specific parameters. In Section 3 we show a number of subscription examples to compare Boolean and top-k/w subscriptions. Section 4 analyzes routing algorithms found in distributed publish/subscribe systems and adapts them according to the requirements of top-k/w processing. In Section 5 we analyze results of an experimental evaluation using synthetic and real data sets which investigates properties of the top-k/w model and present a quantitative evaluation of the routing strategies. A survey of related work in the field is given in Section 6, and Section 7 concludes the paper.

2. Specification of Boolean and Top-k/w Systems

In a typical publish/subscribe system *publishers* publish content (i.e. data objects) which they want to distribute among subscribers in messages called *publications*. *Subscribers* subscribe to the content in which they are interested by issuing messages called *subscriptions*. A *publish/subscribe service*, which can be either *centralized* or *distributed*, is responsible for delivering publications in real-time, i.e. immediately after their publication, to those subscribers whose subscriptions *match* selected publications. A publication matches a subscription when it satisfies all constraints defined by the subscription.

Every publish/subscribe system is distributed in its nature because it consists of a number of clients (i.e. publishers and subscribers) and a publish/subscribe service, which is the mediator that decouples publishers from subscribers. In literature, *centralized publish/subscribe systems* denote systems with a single processing node which performs the matching process, while *distributed publish/subscribe systems* are implemented over a number of network nodes. The processing nodes form an overlay network and distribute the matching load such that each node is responsible for a subset of clients and their subscriptions/publications, and additionally, processing nodes collaborate to forward publications from different parts of the network to all interested subscribers.

In this section we give formal specifications of two different publish/subscribe systems using set theory. The *Boolean publish/subscribe system* is based on the *Boolean matching model*, which is a generalization of the three most common matching models found in literature: topic-based, content-based and type-based models [3]. The matching process uses a given Boolean matching function defined by each subscription. On the contrary, the *top-k/w publish/subscribe system* relies on a novel matching model, named the *top-k/w matching model* which is stateful and time-dependent, because at each point in time t, the parameter k limits the number of matching publications restricting it to the top-k publications published within a subscription window. Each publication is ranked according to a time-independent scoring function, and we assume *time-based windows* in our model. Please note that the specification of the Boolean publish/subscribe system is given for completeness of the paper and is not our original contribution.

2.1. Boolean Publish/Subscribe System

Let $\boldsymbol{B} = (\boldsymbol{N}, \boldsymbol{P}, \boldsymbol{S})$ be a triple, where $\boldsymbol{N}$ is a finite set of nodes, $\boldsymbol{P}$ is a finite set of publications, and $\boldsymbol{S}$ is a finite set of Boolean subscriptions in a Booolean publish/subscribe system. $\boldsymbol{B}$ gives the *structural view* of a Boolean system and determines the boundaries of its state space. A node $n \in \boldsymbol{N}$ may publish publications from $\boldsymbol{P}$, activate or cancel subscriptions from $\boldsymbol{S}$, and perform matching between active subscriptions and publications. Thus, a node may assume one or multiple roles: it may act as a publisher, subscriber, and/or processing node. We refer to a node that activates a subscription as subscriber, and analogously, to a node that publishes a publication as publisher.

Definition 2.1 (Publication)**.** *Let $n_p \in \mathbf{N}$ be a node, let t_p^A be a point in time, and let c_p denote some published content. We define a triple $p = \{c_p, n_p, t_p^A\} \in \mathbf{P}$ as a publication with content c_p that is published by node n_p at a point in time t_p^A.*

The *time of appearance* t_p^A denotes a point in time when publication p appears in the system. We assume publication content is certain, and its content and time of appearance do not change after entering the system. We further assume publications are assigned unique t_p^A when entering the system such that all publications can be ordered by their time of appearance. This is indeed true for a centralized system with a single node assigning unique timestamps to incoming publications, while this assumption does not hold in a distributed setting. However, the assumption can be further relaxed in a distributed setting such that each node assigns unique timestamps to its incoming publications, and thus the model does not require time synchronization between the nodes in the entire system.

Definition 2.2 (Boolean Subscription)**.** *Let $n_s \in \mathbf{N}$ be a node in the system, let t_s^A and t_s^C be two points in time such that $t_s^A < t_s^C$, and let $m_s : \mathbf{P} \mapsto \{\top, \bot\}$ be a time-independent Boolean matching function over $\mathbf{P}$. We define a quadruple $s = \{m_s, n_s, t_s^A, t_s^C\} \in \mathbf{S}$ as a Boolean subscription.*

A Boolean subscription is defined by a Boolean matching function m_s associated with two additional parameters: *time of activation* t_s^A denoting a point in time when subscription s is activated in the system, and its *time of cancellation* t_s^C. We say that the subscription is active within the period $(t_s^A, t_s^C]$. Analogously to publications, we assume that t_s^A and t_s^C are implicit timestamps assigned by the processing nodes. Of course, the transmission of subscriptions and publications to processing nodes introduces certain latency compared to the time of creation at the source nodes. However, we can provide correctness guaranties regarding the matching process by tolerating bounded subscription/publication propagation times, as it is commonly done in distributed implementations. Additionally, it is important to stress that a matching function $m_s(p)$ is time-independent and depends exclusively on the publication content, i.e. $m_s(p) = f(c_p)$. Since both the matching function and object content are static, a Boolean subscription is a *stateless* publication filter.

Definition 2.3 (Matching Publications)**.** *Let $s \in \mathbf{S}$ be a Boolean subscription, and let t be a point in time when s is active. We define the set of matching publications $P_s(t) \subseteq \mathbf{P}$ for s at t as follows:*

$$P_s(t) \stackrel{def}{=} \{p : (p \in \mathbf{P}) \wedge (t_s^C \geq t > t_p^A > t_s^A) \wedge (m_s(p) = \top)\}. \quad (1)$$

The matching function m_s assigns $\top$ to all s's matching publications, while the set of matching publications at a point in time t comprises only those matching publications that have appeared between the activation of s and t. Thus subscriptions reference only future publications, i.e. publications appearing after subscription activation, and cannot reference publications published before subscription activation.

Next, we define the Boolean publish/subscribe system.

Definition 2.4 (Boolean Publish/Subscribe System)**.** *Let* **N** *be a finite set of nodes, let* **P** *be a finite set of publications, and let* **S** *be a finite set of Boolean subscriptions. We define a triple* $\mathbf{B} = (\mathbf{N}, \mathbf{P}, \mathbf{S})$ *as a Boolean publish/subscribe system iff for each Boolean subscription* $s \in \mathbf{S}$*, the system delivers all matching publications to the subscriber issuing s, and each of these publications is delivered exactly once and immediately after being published, while the system does not deliver non-matching publications to the subscriber of s.*

This definition specifies two important properties for real-time system specification [14, 15]: *liveness* and *safety*. The *liveness* property asserts that "all matching publications will eventually be delivered to their subscribers", while the *safety* property asserts that "non-matching publication will never be delivered".

2.2. Top-k/w Publish/Subscribe System

A top-k/w publish/subscribe system (top-k/w system) is defined similarly to the Boolean system as a triple $\boldsymbol{B} = (\boldsymbol{N}, \boldsymbol{P}, \boldsymbol{S})$, where $\boldsymbol{N}$ is a finite set of nodes, $\boldsymbol{P}$ is a finite set of publications, and $\boldsymbol{S}$ is a finite set of top-k/w subscriptions. Similarly to the Boolean system, a node $n \in \boldsymbol{N}$ may publish publications from $\boldsymbol{P}$, and activate or cancel subscriptions from $\boldsymbol{S}$. A publication from the top-k/w system is identical to the one from the Boolean system, see Definition 2.1. However, top-k/w subscriptions are quite different from Boolean subscriptions.

Definition 2.5 (Top-k/w Subscription)**.** *Let* $n_s \in \mathbf{N}$ *be a node in the system, let* t_s^A *and* t_s^C *be two points in time such that* $t_s^A < t_s^C$*. For a given time-independent scoring function* $u_s : \mathbf{P} \mapsto \mathbb{R}$*,* $k_s \in \mathbb{N}$*, and* $w_s \in \mathbb{R}^+$*, we define a sextuple* $s = \{u_s, n_s, t_s^A, t_s^C, k_s, w_s\}$ *as a top-k/w subscription.*

A top-k/w subscription is defined by a scoring function u_s associated with two additional parameters, the query window size w_s (time-based window), and parameter k_s denoting the number of top publications from the window that have to be delivered to subscriber n_s. Furthermore, as top-k/w subscriptions are continuous, they have a predefined time of activation and cancellation, which are implicit timestamps assigned by a nodes. A point in time t_s^A represents the *time of subscription activation*, while t_s^C is the *time of subscription cancellation*. Analogous to the Boolean subscription, we say that the top-k/w subscription is active within the period $(t_s^A, t_s^C]$.

Scoring functions are application specific and their explicit definition depends completely on the application scenario in which the top-k/w system is used, as discussed further in Section 3. Similarly to Boolean matching functions, we assume that scoring functions are time-independent and depend exclusively on publication content, i.e. $u_s(p) = f(c_p)$. Please note that our model supports generic scoring functions because we do not impose any specific constraints on scoring function definition, such as monotonicity, which is frequently assumed by many top-k processing techniques [16]. Our assumption is that publications can be ranked consistently because a tie-breaking criterion can give preference to a more recent object. Scoring functions may assign ranks to publications either in descending or ascending order of their scores. Without loss of generality, in this paper we assume the applied scoring function is such that lower scores are preferable to higher scores, and therefore assign ranks to publications scores in ascending order. Notice that although data objects are static and their scores do not change over time, their ranks may change over time as new publications appear in the system, while previously appeared publications are dropped from a subscription window.

A publication is within the subscription window at a point in time t, if the publication appears in the system after subscription activation, and less than w_S time units have passed since its appearance. This is formally stated by the following definition.

Definition 2.6 (Publications in a Subscription Window)**.** *We define the set of publications in the subscription window of an active subscription s at t as follows:*

$$W_s(t) \stackrel{def}{=} \{p : (p \in \mathbf{P}) \wedge (t_s^C \geq t_p^A + w_S \geq t > t_p^A > t_s^A)\}. \quad (2)$$

Only publications within a subscription window are of interest to a subscription, and thus the model supports *ad-hoc subscriptions referencing future publications*. Next, we define top-k publications from the subscription window at a point in time t.

Definition 2.7 (Top-k/w Publications)**.** *We define the set of top-k publications in the subscription window of an active subscription s at t as follows:*

$$T_s(t) \stackrel{def}{=} \{p : |B_s(t, p)| < k\}, \quad (3)$$

where $B_s(t, p) \stackrel{def}{=} \{p' : (p' \in W_s(t)) \wedge (u_s(p') < u_s(p))\}$.

In other words, a publication p is a top-k publication for a subscription s at t if p is within the current window of s, and there are less than k higher ranked publications than p in this window. Note that the set of top-k/w publications is continuously being updated and changes over time, and that p is delivered to the subscriber issuing s when p becomes an element of T_s for the first time. Since a set of top-k/w publications depends on other publications in the subscription window, top-k/w subscriptions are *stateful* publications filters.

Finally, we define the top-k/w publish/subscribe system.

Definition 2.8 (Top-k/w Publish/Subscribe System)**.** *Let* **N** *be a finite set of nodes, let* **P** *be a finite set of publications, and let* **S** *be a finite set of top-k/w subscriptions. We define a triple* $\mathbf{B} = (\mathbf{N}, \mathbf{P}, \mathbf{S})$ *as a top-k/w publish/subscribe system iff for each top-k/w subscription* $s \in \mathbf{S}$*, the system delivers all top-k/w publications to the subscriber of s, and each of these publications is delivered exactly once and immediately after it becomes an element of* T_s *for the first time, while the system does not deliver non-top-k/w publications to the subscriber of s.*

Again, this definition follows the usual practice in real-time system specification since it defines the *liveness* property asserting that "top-k/w publications will eventually be delivered to their subscribers", while the *safety* property asserts that "non-top-k/w publications will never be delivered".

2.3. Top-k/w Subscription Parameters

Hereafter we discuss how a top-k/w system behaves under specific conditions when its parameters k_s and w_s assume extreme values. We investigate system performance when $k_s \to \infty$, $w_s \to 0$, $w_s \to \infty$, and when the number of publications within the window of size w_s is smaller than k. Finally, we contrast these extreme cases to typical values of these parameters.

Parameter $\mathbf{k_s \to \infty}$. In this setup, each published publication p matches s since an infinite number of publications can be inserted into T_s according to Definition 2.7. As a consequence, every published publication is delivered to subscriber n_s as stated in Definition 2.8. We refer to this extreme setup as the top-∞ case.

Parameter $\mathbf{w_s \to 0}$. This setup is similar to the previous one as each p is a single publication within the window of s at $t_p^A + \epsilon$, where ϵ is an infinitely small amount of time, and thus it becomes an element of T_s according to Definition 2.7. As a consequence, every published publication is delivered to subscriber n_s as stated in Definition 2.8. Therefore, this extreme scenario is also the top-∞ case.

Parameter $\mathbf{w_s \to \infty}$. In this scenario, the window of s is not sliding, but expands in time. Thus only the publications among k_s best ranked publications published from the activation of s are delivered to node n_s. As time passes it is progressively harder and harder for a newly published publications to become a top-k/w publication. We refer to this extreme scenario as the top-k case.

Low intensity of publishing. Let us model the publishing process as a homogeneous Poisson process with parameters $\lambda \leq k_s/w_s$ and $\tau = w_s$, where λ is the expected number of generated publications within a time period τ. Thus the number of newly published publications within the window of size w_s is smaller than k_s and subsequently, according to Definition 2.7, all published publications are delivered to subscriber n_s. Therefore, this extreme scenario is also the top-∞ case.

Typical setup. In a typical scenario, subscription parameters have non-extreme values, while the intensity of publishing is related to subscription parameters as follows: $\lambda > k_s/w_s$. Therefore, such top-k/w subscription acts as a filter for published publications since the number of publications within a sliding window is larger than k_w.

Figure 4 depicts possible parameter values for top-k/w subscriptions, and relates them to the identified special setup cases. It shows that a typical parameter setup is located between the top-k case which delivers top-k publications since subscription activation, and top-∞ case which delivers all incoming publications irrespective of subscription's scoring function.

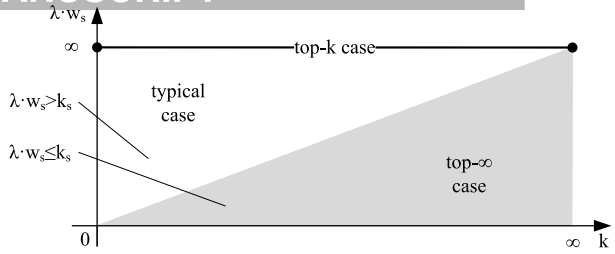


Figure 4: Special cases of top-k/w subscription parameters.

2.4. Comparison of the Boolean and Top-k/w System

In this subsection we show that a top-k/w system is a generalization of a Boolean system, i.e. that every Boolean system is a special case of the corresponding top-k/w system. We can easily achieve that a top-k/w system behaves as a Boolean system by selecting specific values of top-k/w subscription parameters and scoring function, and thus every top-k/w system implementation also supports Boolean subscriptions without further modifications.

Theorem 2.1. *The Boolean system is a special case of the top-k/w system where for every* $s \in \mathbf{S}$ *with parameters* $k_s = 1$ *and* $w_s \to \infty$*, its scoring function* u_s *is defined as follows*

$$u_s(p) \stackrel{def}{=} \begin{cases} 0 & \textit{iff } m_s(p) = \top \textit{ or} \\ 1 & \textit{otherwise,} \end{cases} \tag{4}$$

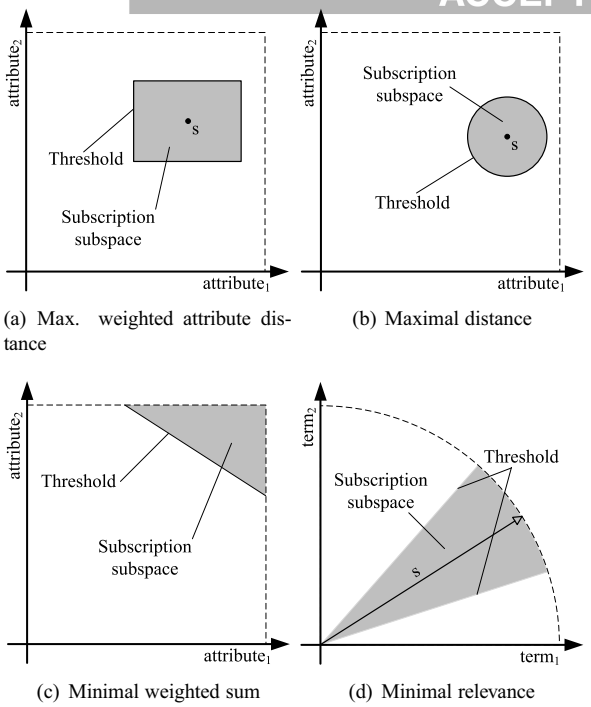
where m_s *is the matching function of* s *in the Boolean system.*

Proof. To prove this theorem we have to show that such a top-k/w system delivers the same set of publications to each subscriber as the corresponding Boolean system. From Definitios 2.3 and 2.7 we see that every publication which matches a subscription in the Boolean system will also match the corresponding subscription in the top-k/w system, because when a publication is published if it matches m_s its score is set to 0, and thus it becomes a top-1 publication at $t_p^A + \epsilon$ while all other publications published since subscription activation are given the score of 1. Therefore, the matching process is time-independent as in the corresponding Boolean system. As the safety properties of both systems forbid delivery of non-matching publications, we have proven the theorem. □

3. Types of Subscriptions in Boolean and Top-k/w Systems

We have defined matching functions of Boolean subscriptions and scoring functions of top-k/w subscriptions quite generally as functions which depend exclusively on publication content. However, for a practical implementation of a publish/subscribe system, efficient implementation of a matching process depends greatly on publication and subscription definition and scoring function. We are particularly interested in distance, aggregation, and relevance functions as scoring functions commonly used in application domains that we envision for the top-k/w model. Thus it is important to analyze different types of Boolean and top-k/w subscriptions to understand the processing tasks of publish/subscribe nodes as the matching process is one of their most prominent tasks.

A typical publication is defined either as a point from an attribute-space or vector in vector-space with static content.



(a) Max. weighted attribute distance (b) Maximal distance

(c) Minimal weighted sum (d) Minimal relevance

Figure 5: Boolean subscriptions with different matching functions.

These are standard assumptions in state-of-the-art content-based publish/subscribe systems. The size of publications is directly influenced by the dimensionality of an attribute or vector space. In practice, attribute spaces usually have low-dimensionality, while vector spaces are usually highly dimensional since their dimensionality is determined by the vocabulary size of unstructured textual collection.

3.1. Boolean Subscriptions

In this paper we focus on content-based Boolean subscriptions as the prevailing type of Boolean subscriptions [3, 17]. A Boolean subscription is defined as a subspace of attribute or vector-space, and we refer to it as the *subscription subspace of interest*. We assume subscription subspace of interest is defined by a matching function m_s which is composed of a scoring function u_s (either an aggregation, distance, or relevance function) and static threshold th. The threshold separates subscription subspace of interest from the remaining attribute or vector-space. A publication with a score that is lower than the threshold is defined as a matching publication for the subscription, i.e. $m_s = \top$ for all publications within subscription subspace of interest, while $m_s = \bot$ otherwise. In other words, the threshold defines the worst publication score that is still considered as matching. Such definition of matching functions is in accordance with the definition of top-k/w scoring functions, and thus allows us to compare the performance of Boolean and top-k/w publish/subscribe systems using the same datasets, which is important for their experimental comparison in Section 5.

The example shown in Figure 5 depicts different Boolean subscriptions with the following matching functions: 1) maximal weighted attribute distance, 2) maximal distance, 3) minimal weighted sum and 4) minimal relevance. For each of these matching functions we depict a threshold that separates a subscription subspace of interest from the rest of the space. In Figure 5(a) we see that the subspace of interest is a hyper-rectangle in the attribute space and that its boundaries represent subscription threshold. It is important to mention that a subspace of interest of a content-based subscription is typically modeled as a logical expression—-conjunction of predicates—-where each predicate defines a simple constraint on an attribute, and may be represented geometrically as a hyper-rectangle [18]. Similarly, in Figure 5(b) the subspace of interest is represented as a hyper-sphere in the attribute space, where hyper-sphere radius represents the subscription threshold. Subscription threshold in Figure 5(c) is a hyper-plane which divides the subspace of interest from the rest of the attribute space. Please note that in this case we prefer larger scores. Finally, the subspace of interest in Figure 5(d) is represented as a cone with a convex base in the vector space. In this case, subscription threshold is calculated as the cosine of the the cone opening angle.

As an example, we show how Boolean matching functions are defined in a two-dimensional attribute and vector spaces. The maximal weighted attribute distance is defined as $m_s = u_s \leq th = max(c_1 \cdot |x_s - x_p|, c_2 \cdot |y_s - y_p|) \leq th$, where ordered pairs (x_s, y_s) and (x_p, y_p) are points representing the subscription and publication, respectively, while c_1, c_2 and th are constants. A typical usage scenario for such subscription is subscribing to ads about apartments for rent by specifying the maximal allowed deviation from characteristics of an "ideal" apartment, which is modeled by (x_s, y_s). Similarly, the maximal distance is defined as $m_s = u_s \leq th = \sqrt{(x_s - x_p)^2 + (y_s - y_p)^2} \leq th$. By issuing such subscriptions we could, for example, subscribe to apartment ads by specifying the cumulative deviation from an "ideal" apartment. In the case of the minimal weighted sum, subscription threshold is a separation line $k_1 \cdot x + k_2 \cdot y = th$, where x is a value of $attribute_1$ and y is a value of $attribute_2$. In this case, the subscription matching function is defined as $m_s = u_s \geq th = (k_1 \cdot x_p + k_2 \cdot y_p) \geq th$. This type of subscription is used for evaluating published objects according to their usefulness for the subscriber. Finally, in the case of two-dimensional vector space, the subscription matching function (i.e. minimal relevance) is defined using the cosine between subscription and publication vectors: $m_s = u_s \leq th = (\overrightarrow{v_s} \cdot \overrightarrow{v_p})/(\|\overrightarrow{v_s}\| \cdot \|\overrightarrow{v_p}\|) \leq th$, where v_s and v_p are subscription and publication vectors. Subscriptions to news articles using keywords are a typical usage scenario for this type of subscriptions.

To conclude, the matching task of publish/subscribe nodes is such that algorithms are needed to organize subscriptions in such structures to efficiently identify a subset of subscriptions whose subspace of interest covers an incoming publication.

3.2. Top-k/w Subscriptions

A publication p within the window of a top-k/w subscription matches a subscription when there are less than k higher ranked publications within the window. This can happen either:

1. at the moment of p's publishing when there are less than

k higher ranked publications within the subscription window, or

2. at a later point in time when one of top-k publications is dropped from the subscription window and *p* has the highest rank among all other publications from the window that are not within top-k publications.

The implementation of the first scenario is quite straightforward: Each published publication is compared against the list of current top-k publications of a subscription, and, in case its rank is higher than the rank of the last top-k publication in the list, it is delivered to the subscriber. This requires continuous maintenance of top-k publications in memory, because, although these publications have already been delivered, they are continuously compared to newly-published publications.

The second scenario requires a more elaborate solution as it has already been recognized in [10, 9, 6, 7]: We need to instantly fill in the slot of a dropped top-k publication with a publication that currently has the best rank among non top-k publications within the query window. Therefore, we need to store additional *candidate publications* in memory that have the potential to become top-k publications in future, which makes the efficient processing of such subscriptions challenging. Consequently, existing top-k/w processing algorithms define candidate publications quite differently:

- The algorithm, presented in [9], uses a set of top-k among *b* most recent publications to filter out objects that cannot become top-k objects in future. This algorithm relies on the *dominance property* which states that only non-dominated publications are eligible candidate publications. A *non-dominated publication* from a subscription window is a publication for which there are less than *k* other publications in a subscription window with a better rank and later time of appearance. Conversely, a dominated publication is a publication for which there are already *k* or more publications in the window that are more recent and with a lower score than the dominated publication, and thus such publication cannot become a top-k/w publication in future. This algorithm defines a subscription threshold as the score of a *k*-th ranked publication among *b* most recently published publications, which are stored in a FIFO buffer and shared among all subscriptions stored at a processing node. We name this algorithm the *Strict candidate pruning Algorithm with Strict Filter* (SASF) because it continuously maintains a set of strictly non-dominated candidate publications in memory, together with an additional set of *b* most recent publications, which is used for filtering of dominated publications.

- Top-k/w processing algorithms (SMA, TMA, SNN and CPM on sliding-windows) presented by Mouratidis at. al in [6, 8] consider all publications within a subscription window as candidates. In the case of the second scenario, these algorithms call the CPM (Conceptual Partitioning Method), an efficient algorithm for continuous nearest neighbour (NN) monitoring, to fill the slot of a dropped top-k publication. Between two consecutive CPM calls, such a top-k/w subscription is only interested in top-k publications, and therefore, its threshold is defined as the score of the *k*-th ranked publication after the most recent CPM call. SMA and SNN call the CPM algorithm less often than TMA and CPM over sliding-windows since the former algorithms additionally store a set of non-dominated candidates in memory, and first try to fill the slot of a dropped top-k publication with the best of them, or call the CPM when this set is empty.

- Finally, a top-k/w subscription that is based on our probabilistic algorithm (PA) presented in [10, 19], is interested only in publications for which the probability to become top-k/w publications in future is larger than a predefined probability σ. As a probabilistic algorithm, it ignores publications with probability smaller than σ and introduces a controllable probability of error (both false positive and false negative top-k/w publications). Therefore, at a point in time, the threshold of a PA-based top-k/w subscription is defined as the score of the publication with the worst score whose probability to become top-k/w publication is larger than σ.

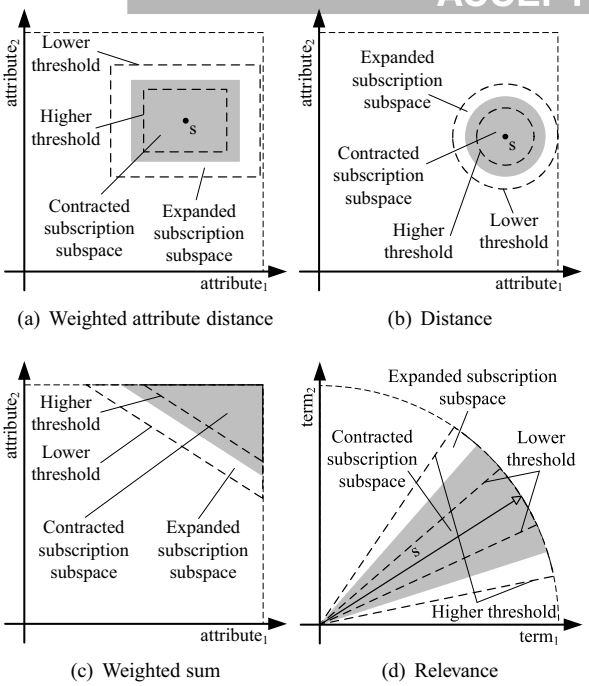
For our subsequent discussion it is important to understand that the current threshold value of a top-k/w subscription is directly related to the set of top-k and candidate publications, regardless of the used top-k/w processing algorithm. Additionally, to calculate the threshold, we need to know all elements of both sets. Moreover, since the sets change very frequently, this also reflects on subscription thresholds. Therefore, while Boolean subscriptions define static subspaces of interest, top-k/w subscriptions are interested in dynamically changing subspaces of the attribute space.

Analogous to Boolean subscriptions in Figure 5, Figure 6 shows subspaces of interest for four different top-k/w subscriptions defined with the following scoring functions: 1) weighted attribute distance, 2) distance, 3) weighted sum and 4) relevance. For each of these scoring functions we can see a dynamic threshold that separates a subscription subspace of interest from the rest of the available attribute or vector-space. As shown in the figure, if this threshold changes, the corresponding subscription subspace of interest also changes, i.e. it either expands or contracts.

To conclude, a top-k/w publish/subscribe node needs to maintain a set of top-k and candidate objects in memory per subscription, and apply one of the aforementioned algorithms for maintaining those sets and calculating the threshold. The threshold is compared with the score of a new publication which enables the processing node to quickly identify weather the publication is a candidate or not. Note also that each change of the top-k or candidate sets results in threshold change.

3.3. Subscription Indexing

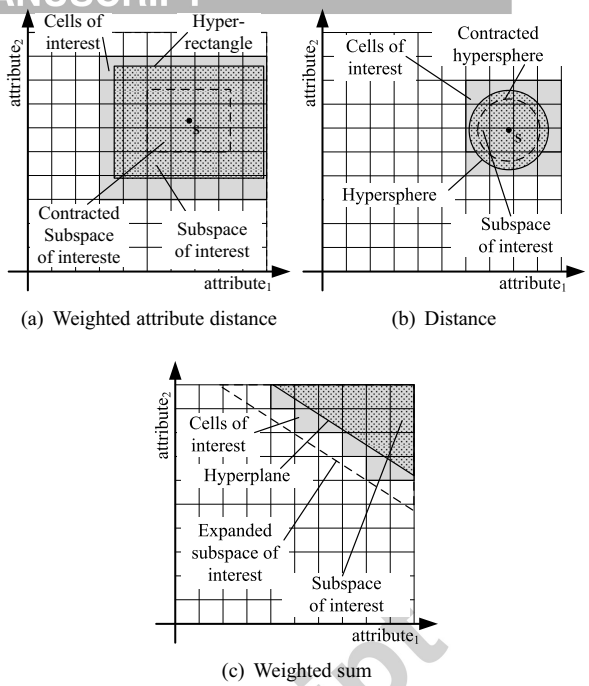
Subscription indexing reduces the number of publications that a Boolean or top-k/w subscription needs to process by identifying and neglecting those publications which are certainly not of interest for a subscription. Hereafter we sketch indexing techniques for distance and aggregation scoring functions.



(a) Weighted attribute distance (b) Distance

(c) Weighted sum (d) Relevance

Figure 6: Top-k/w subscriptions with different scoring functions.



(a) Weighted attribute distance (b) Distance

(c) Weighted sum

Figure 7: Indexing of subscriptions for various scoring functions.

Distance and aggregation (i.e. weighted sum) functions are applied to structured publications represented as points in a multidimensional attribute space. In existing approaches [8, 6, 9], a regular grid is used to index subscriptions in such space. A regular grid divides an attribute space in cells of equal size, while a subscription threshold defines the subscription subspace of interest as shown in Figure 7. Usually, each cell contains a list of subscriptions whose subspace of interest completely or at least partially covers the cell. This makes the processing of incoming publications more efficient since only interested subscriptions are informed about a publication appearing in the cell. Please note that the subspace of interest is covered by the *cells of interest* from the regular grid that encompass a larger portion of the attribute space than the subspace of interest itself. When the subscription subspace of interest changes (i.e. expands or contracts) this can reflect as a change in the corresponding cells of interest. Since this situation happens less often than a regular threshold change, subscription cells of interest change less often than its threshold which is important for a distributed setup where processing node need to exchange threshold updates or changes of subscription cells of interest.

To our knowledge, the indexing of top-k/w queries in vector-space is still an open research problem [20, 21] and is not discussed further in this paper.

4. Routing in Distributed Publish/Subscribe Systems

The core mechanism behind a distributed publish/subscribe service is routing of publications and subscriptions between the nodes in an overlay network. The main issue with routing algorithms is their scalability, which requires careful balancing of the number of propagated messages in an overlay network, and the amount of stored routing information at overlay nodes. The routing information is maintained by all processing nodes: *routing tables* map node identifiers (usually identifiers of their neighboring nodes) to proxy subscriptions received from these nodes. *Proxy subscriptions* are representatives of original subscriptions activated by subscriber nodes, and in this section we contrast Boolean and top-k/w proxy subscriptions to stress statefulness of top-k/w proxy subscriptions. This property largely affects the routing strategies for distributed top-k/w publish/subscribe systems, as we show when analyzing potential routing strategies for such systems. Our original contribution is the adaptation of existing routing strategies commonly found in distributed Boolean publish/subscribe systems to support stateful subscriptions, such as parametrized subscriptions [22] and our original top-k/w subscriptions. For details on routing strategies found in Boolean publish/subscribe systems we refer an interested reader to [23].

4.1. Proxy Subscriptions

Suppose that a client activates a new subscription and sends it to the processing node through which it is connected to a publish/subscribe system. Obviously, this node is aware of the client's subscription and stores it in memory for further processing. However, other nodes should have a copy of the original subscription to decide whether a publication published at another part of the network is of interest for the subscription. We refer to such subscription copies as *proxy subscriptions* since they represent original subscriptions at some other nodes different from the originating node at which the subscription has firstly been activated. In this subsection we compare Boolean

and top-k/w subscriptions and their proxy subscriptions.

4.1.1. Boolean Proxy Subscriptions

Every Boolean subscription is defined by a matching function, subscriber generating the subscription, and activity period. Since such subscriptions are stateless and do not change over time, all proxy Boolean subscriptions are identical copies (i.e. replicas) of the original subscription. Therefore, once such a replica is stored on a node, it remains consistent with its original subscription until subscription cancellation when such replica has to be removed from the node. In the rest of this paper we will not make any difference between a Boolean subscription and its proxy subscriptions since they are identical.

4.1.2. Top-k/w Proxy Subscriptions

A completely different situation occurs in the case of top-k/w subscriptions. Such subscriptions are stateful, and thus their proxy subscriptions need to be in a consistent state with their original subscriptions. From the discussion in the previous section we know the state of a top-k/w subscription is determined by the dynamic set of its top-k and candidate publications. Therefore, a naive approach would continuously ship all top-k and candidate publications to all proxy subscriptions such that they are consistent with the original subscription, and thus enable the node to determine the current threshold value. However, such solution is very impractical because it stores a lot of redundant data in the network and generates huge network traffic. Let us analyze what is the minimal quantity of information needed to decide whether a publications is of interest to a top-k/w subscription or not. Obviously, we need to know the scoring function to calculate publication scores. Then we also need to know the current value of a subscription threshold to identify the current subspace of interest. In other words, to decide whether a publication is of interest to a top-k/w subscription, a processing node needs to know the subscription's scoring function and receive continuous threshold updates. It does not need to know the values of subscription parameters k_s and w_s, nor the elements of its top-k and candidate publication set if top-k/w subscriptions are indexable with their thresholds.

Since scoring functions are static (i.e. time-independent), we only need to synchronize thresholds to make top-k/w proxy subscriptions consistent with their originals. For this reason, we can say that a top-k/w subscription proxy is equal to a parameterized subscription which introduces a threshold as its only parameter [22]. Due to the similarity between parameterized and top-k/w subscriptions, the following discussion investigating the suitability of different routing strategies for top-k/w subscriptions also applies to parameterized subscriptions.

It is important to notice that the process of publication matching against a top-k/w proxy subscription is computationally less intensive than the process of publication matching against the original top-k/w subscription because the former process requires only score calculation and score comparison to subscription threshold. Additionally, it requires less memory since top-k/w proxy subscriptions do not maintain top-k and candidate publications at proxy subscription nodes: Publications with scores below the threshold are sent to nodes that process original subscriptions and maintain subscription top-k and candidate publications.

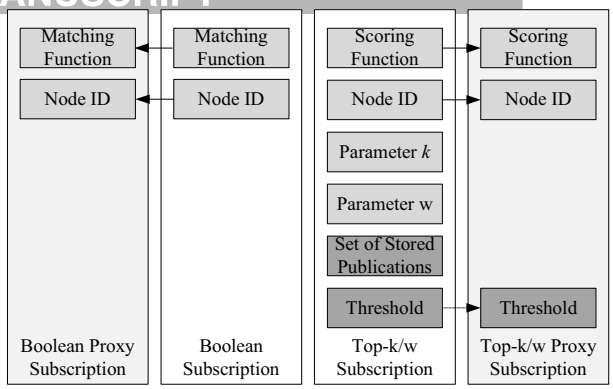


Figure 8: Components of the Boolean, top-k/w subscription and their proxy subscriptions.

Figure 8 shows different components of Boolean, top-k/w, and the corresponding proxy subscriptions. A Boolean proxy subscription is identical to its original since both of them require two pieces of information: a static Boolean matching function and subscriber identifier. A top-k/w subscription is composed of six pieces of information: a static scoring function, subscriber identifier, parameters k_s and w_s, dynamic threshold, and dynamic set of stored publications. A top-k/w proxy subscription requires only the following components: a scoring function, subscriber identifier and dynamic threshold.

4.2. Routing Strategies

According to [23], the commonly used routing strategies in distributed publish/subscribe systems can be grouped into the following categories: flooding, selective routing and gossiping. Furthermore, each of the three categories includes two commonly used strategies, which results in six strategies in total.

Flooding can be implemented as either subscription or publication flooding. This type of routing causes large message overhead since the overlay network is frequently flooded with messages containing either subscriptions or publications.

Selective routing is further classified as covering-based[2]and rendezvous routing. This type of routing tries to reduce message overhead in an overlay network as much as possible. Selective routing is based on the following two observations: 1) subscribers are usually interested in a small portion of published publications and 2) subscriber interests usually overlap. When this is not true, selective routing leads to flooding of an overlay network with either subscriptions or publications.

Gossiping algorithms used in unstructured peer-to-peer overlay networks can further be categorized as basic gossiping, which is purely probabilistic, and informed gossiping, which is partially probabilistic and partially deterministic. These strategies are best suited to systems with a high churn rate, i.e., with a

[2]The publish/subscribe architectural model from [23] considers filtering-based instead of covering-based routing. However, covering-based routing is an advanced version of filtering-based routing, and is more often used in practice.

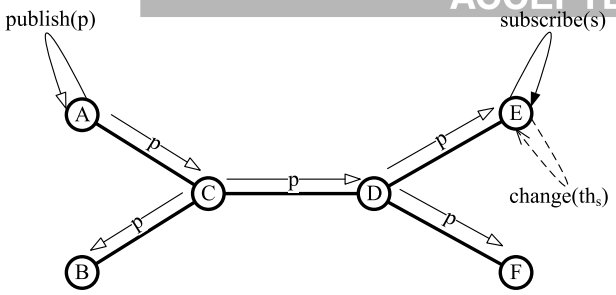


Figure 9: Sequence of events in a distributed top-k/w system with publication flooding.

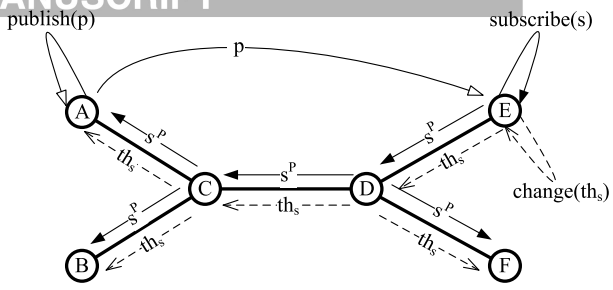


Figure 10: Sequence of events in a distributed top-k/w system with subscription flooding.

highly-dynamic peer-to-peer network topology, to improve the reliability of publication delivery.

The task of adapting existing strategies to specific requirements of top-k/w systems is non-trivial, since Boolean subscriptions and their proxy subscriptions are stateless, while top-k/w subscriptions and their proxies are stateful. Please note that in all routing strategies there is a single node which processes the original top-k/w subscription and maintains the set of top-k and candidate publications, while all other nodes may only be responsible for top-k/w proxy subscriptions with a varying threshold. Also note that in the following examples we assume each node takes all three possible roles (publisher, subscriber, processor).

4.2.1. Publication Flooding

In a distributed Boolean system with publication flooding, subscriptions are stored at subscriber nodes and are therefore not propagated further into the overlay network. When a publication is published, the overlay network is completely flooded with it. Upon receiving a newly published content, each node first forwards this publication to its neighbors, i.e. further into the overlay network, and after that begins the process of matching between this publication and its own Boolean subscriptions to check if this publication matches any of them. Obviously, this strategy is best suited to situations when the majority of nodes is interested in most of the published publications. However, if the number of publishers or publishing rate is high, the overlay network is overloaded with published publications.

In a distributed top-k/w system which employs this routing strategy we do not need to store top-k/w proxy subscriptions in the network since the matching is performed at the subscriber node which also maintains subscription's top-k and candidate publications. For example, in Figure 9 we show a sequence of events in a distributed top-k/w system with publications flooding which consists of 6 nodes. When node E subscribes to s, s is not further propagated into the network. When node A publishes p, the entire network is flooded with p. Furthermore, information about s's threshold change (method change(th_s)) is retained locally. Since this routing strategy does not introduce any additional communication overhead when compared to distributed Boolean systems, we conclude that it is suitable for implementing distributed top-k/w systems, although it suffers from the same drawbacks as publication flooding in Boolean systems.

4.2.2. Subscription Flooding

In a distributed Boolean system with subscription flooding, the overlay network is flooded with every new subscription activation or cancellation. In this routing strategy, publishers match publications they publish against all active Boolean subscriptions in the system. Besides that, they are responsible for direct forwarding (i.e. using a network layer protocol) of matching publications to subscribers. This strategy is best suited to environments that share the processing load (due the matching) in situations where the majority of nodes are also publishers. However, if the frequency of subscription updates is large, the overlay network is overloaded with subscription updates.

When this routing strategy is used in distributed top-k/w systems, each publisher node stores proxy subscriptions of all active top-k/w subscriptions. Furthermore, each subscriber must process the stream of received publications for its own top-k/w subscriptions to set the correct values of their thresholds. When a subscription threshold changes, all nodes in the overlay network have to be informed about this change to update the corresponding top-k/w proxy subscriptions. For example, Figure 10 shows a sequence of events in a distributed top-k/w system with subscription flooding which consists of 6 nodes. When node E subscribes to s, or when the threshold of E's subscription changes, the whole network is flooded with this information. Therefore, we conclude that this routing strategy can be used for distributed top-k/w systems, but is not adequate for stateful subscriptions since the synchronization of subscription thresholds introduces a large communication overhead when compared to distributed Boolean systems. Please note that A's publication p is a top-k/w publication for s since its score is better than or equal to subscription threshold, and therefore p is forwarded directly from A to E since E is the subscription originator which maintains s's top-k and candidate publications.

4.2.3. Covering-based Routing

Covering-based routing in distributed Boolean systems is an extension of subscription flooding which tries to reduce the number of propagated proxy subscriptions by relying on the coverage property between subscriptions. A Boolean subscription is *covered* by another Boolean subscription if every publication which matches the covered subscription also matches

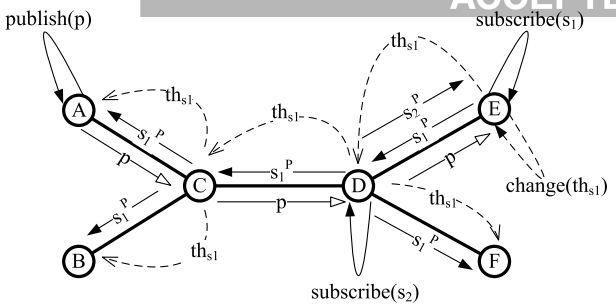


Figure 11: Sequence of events in a distributed top-k/w system with covering-based routing.

the covering subscription, while th opposite does not hold. In case the interests of subscribers do not overlap, this strategy is reduced to subscription flooding and the overlay network is overloaded with subscription updates.

With covering-based routing, a subscription is propagated further through an overlay network only if it is not covered by a previously propagated and still active subscription. Otherwise, a processing node stores it in the list of covered subscriptions, and stops its further propagation. Additionally, processing nodes maintain routing tables mapping identifiers of their neighbors to all uncovered proxy subscriptions received from them. A new publication is matched to proxy subscriptions at all nodes on the reverse path from the publisher to the subscriber. In other words, the publisher forwards this publication to those neighbors from which it has previously received matching proxy subscriptions, and each of the neighbors repeats the process and forwards the publication to its neighbors with matching subscriptions. The process stops when there are no matching subscriptions stored at a node.

When a new subscription is activated in a distributed top-k/w system with covering-based routing, the overlay network is flooded with the corresponding proxy subscription. However, if a proxy subscription is covered by one or more of the previously propagated proxy subscriptions, its propagation is stopped. A top-k/w proxy subscription is *covered* by another top-k/w proxy subscription if, at a point in time, the subspace of interest of the covering subscription contains the subspace of interest of the covered subscription.

For example, Figure 11 shows a sequence of events in a distributed top-k/w system with covering-based routing which consists of 6 nodes. When node E defines a new subscription, the network is flooded by the s_1's proxy subscription (s_1^P). Later, when node D subscribes, but s_2's proxy subscription (s_2^P) is sent only to node E since s_2^P is covered by s_1^P. When a subscription is canceled or its threshold changes, all nodes which store its proxy subscription should delete them or update their thresholds, respectively, and additionally check the covering relations between all proxy subscriptions they store. However, the latter process is non-trivial and may produce large processing overhead at nodes in an overlay network. In other words, not only does this routing strategy flood the overlay network with frequent threshold changes, but it additionally requires recomputation of subscription covering after each threshold change. For this reason, we conclude that this routing strategy can be applied in distributed top-k/w systems, but is not well suited since the synchronization of subscription thresholds introduces a large processing and communication overhead when compared to distributed Boolean systems. Please note that merging[3] of top-k/w subscription proxies can be used with this routing strategy to reduce the routing information stored at nodes, but this would introduce an additional processing overhead in a distributed top-k/w system.

4.2.4. Rendezvous Routing

In every distributed publish/subscribe system with rendezvous routing, the following two methods exists: 1) a method which maps each subscription to an overlay node and 2) a method which maps each publication to an overlay node. Such a mapping is usually achieved by dividing the attribute space to finite number of subspaces, and then by assigning each subspace to a different overlay node. In this way publications matching a subscription are mapped (i.e. assigned) to the same overlay node as subscription. Both mapping methods are typically implemented on top of a structured peer-to-peer network which is then responsible for routing of publications and subscriptions to nodes to which they are mapped. A node to which a publication or subscription is mapped to is called the *rendezvous node* for that publication or subscription. When a new subscription is activated, the subscriber node forwards the subscription augmented by its identifier through the overlay to the subscription's rendezvous node. Similarly, when a publication is published, the publisher forwards the publication through the overlay to the publication's rendezvous node. When a rendezvous node receives an incoming publication, it matches the publication to previously received subscriptions. In this way, each rendezvous node takes responsibility for storing of subscriptions that are mapped to it and matching of them with incoming publications that are also mapped to it. In addition, it is also responsible to deliver matching publications to their subscribers directly (i.e. using a network layer protocol). This strategy is best suited to situations when we want to balance the processing and memory load among overlay nodes. Additionally, when this routing strategy is used, no redundant matching needs be performed in the system, which is a big advantage when compared to covering-based routing.

When this routing strategy is used in distributed top-k/w systems, we do not need to store proxy subscriptions in the network since subscriptions are stored only at their rendezvous nodes that also maintain subscription's top-k and candidate publications. For example, Figure 12 shows a sequence of events in a distributed top-k/w system with rendezvous routing which consists of 6 nodes. We see that E's subscription is routed to node B which is the rendezvous node for this subscription. Later on, when node B receives A's publication p which is mapped to it, it performs top-k/w matching of p and delivers it directly to E because p is a top-k/w publication for E's subscription

[3] At a point in time, two or more top-k/w proxy subscriptions can be *merged* together in a broader subscription which then covers all of the original subscriptions.

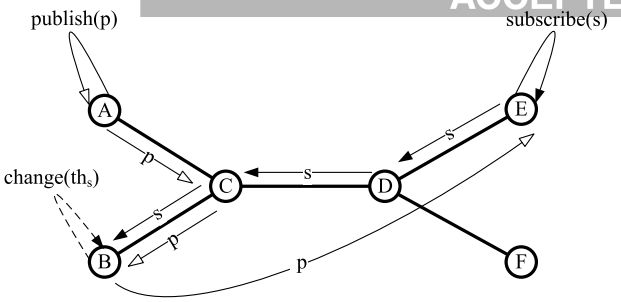


Figure 12: Sequence of events in a distributed top-k/w system with rendezvous routing.

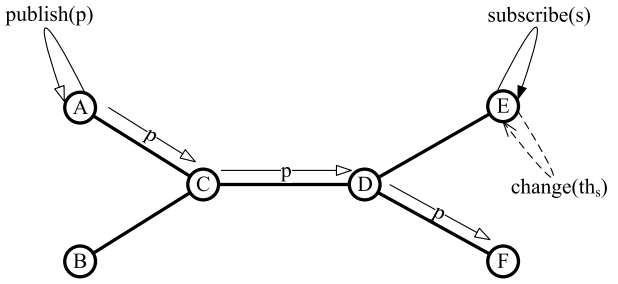


Figure 13: Sequence of events in a distributed top-k/w system with basic gossiping.

s. Similarly, as for publication flooding, a threshold change of E's subscription is retained locally at the rendezvous node B. We conclude that this routing strategy is well suited for distributed top-k/w systems because it does not introduce[4]any additional communication overhead when compared to distributed Boolean systems.

4.2.5. Basic Gossiping

Basic gossiping, as a routing strategy in distributed Boolean publish/subscribe systems, is similar to publication flooding since subscriptions are stored at subscriber nodes and are not propagated further into the overlay network. When a publication is published, it is randomly spread through an overlay network as a gossip. More precisely, in each round of publication spreading, one or a few nodes that have received this publication previously, spread it further to some other neighbors chosen at random. Upon receiving a newly published publication, a node matches the publication against its own subscriptions, and after that randomly choses one or a few of its neighbors and forwards the publication to them. Publication spreading through an overlay network stops after a predefined number of rounds. This routing algorithm is probabilistic and thus does not guarantee publication delivery to all interested subscribers.

When this routing strategy is used in distributed top-k/w systems, similarly to subscription flooding, we do not need to store

[4]However, depending on the applied scoring function and actual implementation of this routing strategy, if rendezvous nodes have to contact other nodes in the overlay network due to the processing of publications, additional communication overhead can be generated, but this has to be investigated on a case by case basis.

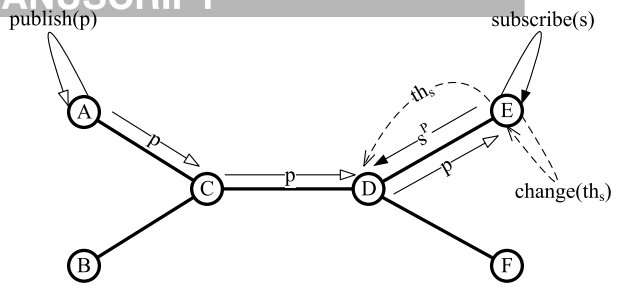


Figure 14: Sequence of events in a distributed top-k/w system with informed gossiping.

proxy subscriptions in the network since the matching is performed at the subscriber side. For example, Figure 13 shows a sequence of events in a distributed top-k/w system with basic gossiping which consists of 6 nodes. We see that E's subscription is retained locally. When node A publishes *p* which is a top-k/w publication for *s*, node E does not receive *p* because it has not spread to E. Similarly to publication flooding and rendezvous routing, a threshold change of E's subscription is retained locally. We conclude that this routing strategy is well suited for distributed top-k/w systems because it does not introduce any additional communication overhead when compared to distributed Boolean systems, but it also cannot provide any guaranties regarding publication delivery.

4.2.6. Informed Gossiping

Informed gossiping, as a routing strategy in distributed Boolean systems, is similar to basic gossiping since both strategies are probabilistic. For both routing strategies, nodes store their own subscriptions, but in case of informed gossiping each node additionally stores subscriptions of its close neighbors. In addition to matching of an incoming publication to its own subscriptions, each node also processes the subscriptions of its neighbors. This is the deterministic part of publication spreading which is followed by the probabilistic part when the node forwards the publication to a randomly chosen set of its neighbors. Similar to basic gossiping, the spreading of a publication through the overlay network stops after a selected number of probabilistic rounds. Therefore, while basic gossiping is purely probabilistic, informed gossiping is partially probabilistic and partially deterministic, and the probability of successful delivery of a publication to subscribers is increased when compared to basic gossiping.

When this strategy is used in distributed top-k/w systems, each node processes the stream of incoming publications for its own top-k/w subscriptions and proxy subscriptions of its close neighbors. For example, Figure 14 shows the sequence of events in a distributed top-k/w system with informed gossiping which consists of 6 nodes. We can see that node D, which is a neighbor of node E, stores its subscription. Proxy subscriptions stored at neighboring nodes have to be synchronized with their originals. For example, node E has to inform D about every threshold change of its subscription. We conclude that this routing strategy is well suited for distributed top-k/w systems since the information about a threshold change is forwarded

only locally. However, when compared to a distributed Boolean system, additional (low and controllable) message overhead exists due to the local synchronization of subscription thresholds between neighboring nodes.

To conclude, the following strategies are well suited for distributed top-k/w systems since they introduce marginal communication overhead when compared to distributed Boolean systems: publication flooding, rendezvous routing, basic gossiping, and informed gossiping. However, please note that publication flooding can overload the network with publications in case of high publication rates. Basic gossiping is probabilistic and cannot provide delivery guaranties, while informed gossiping has higher delivery probability at the expense of an increased message overhead. This analysis brings us to the conclusion that the most suitable routing strategy for distributed top-k/w publish/subscribe systems is rendezvous routing.

However, please note that defining the pair of methods that map publications and subscriptions to rendezvous nodes is a non-trivial task. Especially since a subscription mapping function has to intersect with a publication mapping function such that publications which match a subscription are mapped to the same rendezvous node as subscription. This implies clustering of the attribute space, which is not always possible in practice. An example of mapping n-dimensional attribute space onto a CAN overlay network is given in [13].

5. Experimental Evaluation

We start this section by presenting an experimental study based on a prototype implementation, comparing Boolean and top-k/w subscriptions to examine model properties. In particular, we investigate the number of delivered publications per subscription, and to check experimentally how the models adapt to different publication distributions and publication intensity. Additionally, to experimentally examine the frequency of top-k/w subscriptions updates which influences the number of generated messages in distributed top-k/w systems, we compare the performance of two top-k/w subscription processing algorithms, PA and SASF, previously mentioned in Section 3.2. We find the algorithms from [6, 8] costly for a distributed environment since they perform periodical CPM calls for nearest neighbor discovery which could generate too much network traffic.

In the second part this section, we use an analytical evaluation to compare the number of exchanged messages for different routing strategies in distributed Boolean and top-k/w systems. As previously explained in Section 4, the impact on the performance of a publish/subscribe system due to the introduction of top-k/w subscriptions is twofold: 1) additional processing and memory may be required to handle subscriptions and 2) the communication cost between nodes is increased. In this evaluation, we focus on the latter aspect, without entering into details about the efficiency of matching and indexing algorithms that run at the processing nodes.

5.1. Evaluation of Boolean and Top-k/w Matching Models

We have selected sliding-window k-nearest neighbors (k-NN) subscriptions as a use case for our evaluation since these subscriptions are regarded as one of the most prominent top-k/w problems. Both subscriptions and publications in our experiments are represented as points in a d-dimensional Euclidean space. The score of a publication p with respect to a subscription s is calculated as: $u_s(p) = distance(point_p, point_s) = [\sum_{i=1}^{d}(v_i - v_i)^2]^{\frac{1}{2}}$, where $point_p = \{v_1, v_2, \ldots, v_d\}$ and $point_s = \{v_1, v_2, \ldots, v_d\}$ are points representing the publication p and subscription s, respectively. Obviously, this scoring function prefers lower scores to higher scores.

Table 1: Default values of parameters used in the model evaluation.

Parameter	Symbol	Value
Number of publications	P	10^6
Number of subscriptions	S	400
Intensity of object appearance (objects/min)	λ	1000
Window size (time-based) in minutes	w	40
No. of top-k objects	k	9
Data dimensionality	d	4
PA: probability of error	σ	10^{-3}
SASF: size of recent buffer	b	2000

The experimental evaluation is performed using two synthetic and one real data set. In particular, we generated uniform and clustered Gaussian data within the interval [0, 1]. The clustered Gaussian data had two randomly chosen cluster centers and variance equal to 0.1 for each dimension. It has similar properties to the distribution of our real data set which is an excerpt of the LUCE deployment data, environmental data collected from a large-scale wireless sensor network deployed within the project SensorScope[5]. The LUCE deployment data is preprocessed to extract 4-dimensional data objects (solar panel current, global current, primary buffer voltage and secondary buffer voltage) and normalized to the values within the interval [0, 1].

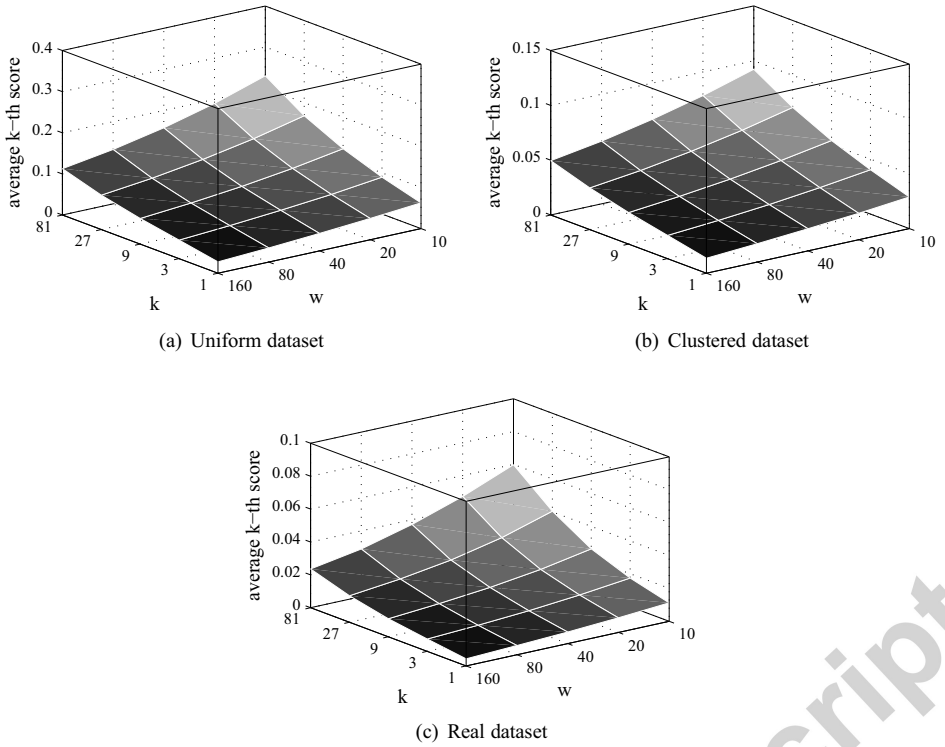
The default scenario used in this experimental evaluation is the following: First we generated the set of subscriptions, either by taking a random sample from the LUCE deployment data, or by generating subscriptions using one of the listed distributions. Second we generated the publications, either by randomly choosing publications from the LUCE deployment data, or by generating publications using the same distribution as for the previously generated queries. Finally, after P publications, we analyzed the obtained results. The default simulation parameters used in experiments are specified in Table 1.

In the following we first examine the observed average threshold values and average top-k scores. After that we observe the average number of delivered publications for Boolean and top-k/w subscriptions, and examine the number of top-k/w subscription changes for different top-k/w implementations.

5.1.1. Top-k Scores of Top-k/w Subscriptions

In the first experiment we analyze the average top-k score (i.e. score of the k-th ranked referenced publication) of top-k/w

[5] http://sensorscope.epfl.ch/



(a) Uniform dataset

(b) Clustered dataset

(c) Real dataset

Figure 15: Average top-k score of a top-k/w subscription for different datasets.

subscriptions for different datasets. The purpose of this experiment is to identify the values of Boolean subscription thresholds which we will later use for the comparison of Boolean and top-k/w subscriptions. If the threshold of a Boolean subscription is equal to the average top-k score of a top-k/w subscription, these two subscriptions will have a similar number of matching publications. Figure 15 shows our results expressed as the average score of the k-th ranked publication. Please note that the scales on the y-axes are different for different datasets. We can see that the average top-k scores are different for different datasets, which is expected since such scores have to be lower for more clustered datasets, and that these scores increase with increasing parameter k and decreasing parameter w. The shapes of the average top-k scores are similar for different datasets which implies that top-k/w subscriptions adapt well to a given dataset.

5.1.2. Number of Matching Publications

In the second experiment we analyze the average number of matching publications per subscription for Boolean and top-k/w subscriptions while varying the publication intensity λ. In this experiment we are using only the uniform dataset and the default values of all parameters specified in Table 1. The obtained simulation results are shown in Figure 16. We can see that similarly to the previous experiment, and due to the adaptability of the top-k/w model to different publication intensities, the average number of matching publications per top-k/w subscription does not depend on the parameter λ, while for Boolean subscriptions, on the contrary, this number heavily depends on the parameter λ since it increases linearly with λ.

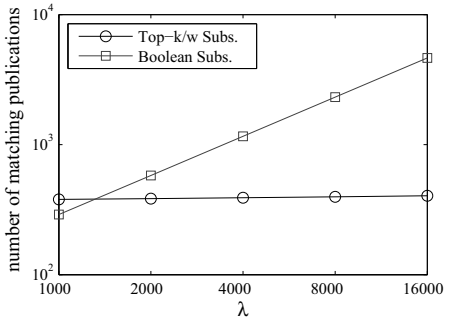
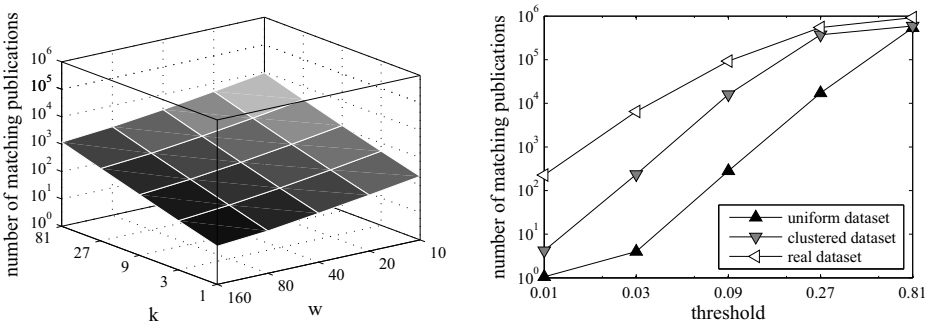


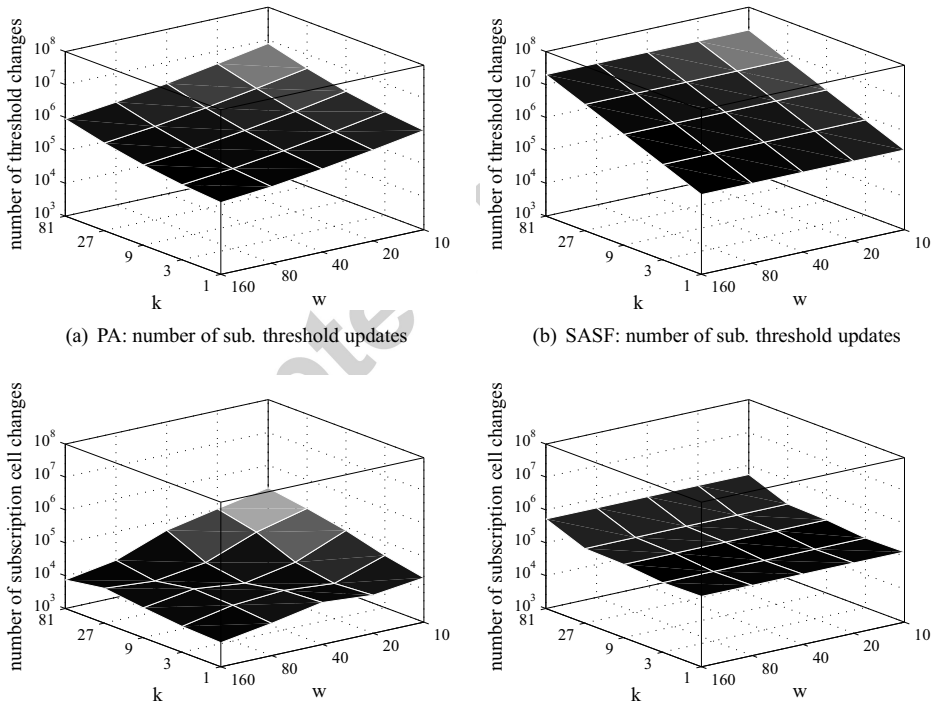
Figure 16: Number of matching publications per Boolean and top-k/w subscription.

The setup of the third experiment is similar to the second experiment since we analyze the average number of matching (i.e. delivered) publications per subscription for Boolean and top-k/w subscriptions while varying subscription parameters k and w. Please note that due to the adaptability of top-k/w subscriptions to different publication datasets, the average number of matching publications per top-k/w subscriptions does not depend on the dataset. This evaluation, shown in Figure 17(a), experimentally proves that the top-k/w model performs effective filtering regardless of the dataset. As expected, when increasing w, the number of matching publications decreases, while it increases when increasing k. On the contrary, the number of matching publications for Boolean subscriptions heavily depends on the dataset (and selected threshold value). As we can



(a) Top-k/w subscription: all datasets (b) Boolean subscription: all datasets

Figure 17: Average number of matching publications per Boolean and top-k/w subscription.



(a) PA: number of sub. threshold updates (b) SASF: number of sub. threshold updates

(c) PA: number of sub. cell updates (d) SASF: number of sub. cell updates

Figure 18: Number of subscription threshold and cell updates for different types of top-k/w subscriptions.

see in Figure 17(b), a small change in the Boolean subscription threshold results in a large variation of the number of matching publications for different datasets. This shows that Boolean subscriptions do not provide any means for controlling the number of delivered publications per subscription. Please note that the Boolean threshold value of $th = 0.09$ is analogous to the average top-k score of top-k/w subscription with parameters $k = 9$ and $w = 40$ for the uniform dataset as shown in Figure 15(a), and these subscriptions thus have similar numbers of matching publications for these values of parameters.

5.1.3. Number of Threshold Updates for Top-k/w Subscriptions

In the fourth experiment, we investigate the average number of threshold updates caused by top-k/w subscriptions and the effect of indexing using regular grid. We analyze two approaches for informing other nodes about subscription updates. The first approach generates threshold change messages upon each threshold change, while the second divides the Euclidean space to cells of equal size (regular grid), and informs other nodes only when subscription's cells of interest expand or contract. The number of subscription threshold and cell changes are shown in Figure 18. We can see that the number of updates is much lower in the case of the latter approach, especially for the PA-based implementation, while the former should be avoided due high frequency of threshold updates. PA-based implementation generates less cell updates then SASF and is preferable in distributed setup if an application can tolerate the error introduced by PA. This number varies from 10^3 to 10^4 cell updates (for PA) which is reasonable when processing $S = 400$ subscriptions and $P = 10^6$ processed publications.

5.2. Evaluation of Routing Strategies in Distributed Boolean and Top-k/w Systems

The default hypothetical scenario used in this analytical evaluation is similar to the default scenario used in the experimental evaluation of the models. In this scenario we first activate the set of subscriptions, then publish the set of publications, and in the end estimate the total number of exchanged messages in the case of Boolean and top-k/w subscriptions. The messages which are exchanged in the system during the scenario are related to either activation of subscriptions, publishing of publications, delivery of matching (i.e. top-k/w publications) or updating of top-k/w proxy subscriptions (i.e. their thresholds). The default parameters used in this evaluation are specified in Table 2. The values of these parameters are either obtained by the evaluation of the models or are equal to the values which were considered in the evaluation. As we can see, the number of top-k/w subscription updates U is equal to 10000, which corresponds to the number of cell updates for PA-based subscriptions with parameters $k = 9$ and $w = 40$. Similarly, the number of matching publications P is equal to $385 \cdot S$ since this number is obtained by the previous simulation for Boolean subscriptions with threshold $th = 0.09$ and top-k/w subscriptions with parameters $k = 9$ and $w = 40$. Table 3 shows the estimated number of exchanged messages for different types of subscriptions and routing strategies. Hereafter we explain and analyze these estimated numbers of exchanged messages.

Table 2: Default values of parameters used in the evaluation of routing strategies.

Parameter	Symbol	Value
Number of publications	P	10^6
Number of subscriptions	S	400
Number of nodes	N	256
Number of top-k/w subscription updates	U	10000
Number of matching publications	M	$385 \cdot S$
covering probability	cp	0.5
spreading probability	sp	0.1
node degree	nd	3

Publication Flooding. As we can see in Table 3, the number of exchanged messages in distributed Boolean and top-k/w system with publication flooding is the same. In this routing strategy, the matching is performed by subscriber nodes, and thus each published publication has to be delivered to all other nodes in the system, which results in $(N-1) \cdot P$ messages. Therefore, there is no increase in the total number of exchanged messages in the top-k/w system when compared to the Boolean system.

Subscription Flooding. In the case of subscription flooding, every activated subscription has to be delivered to every other node in the system. Furthermore, matching is preformed by the publisher node such that matching publications can be directly delivered, by a network layer protocol, to their subscribers. The total number of exchanged messages in the Boolean system is equal to $S \cdot (N-1) + M$. In the top-k/w system, the information about each threshold update has to also be delivered to all other nodes which results in $S \cdot (N-1) + M + U \cdot (N-1)$ messages, i.e. the percent increase is huge (996%).

Covering-based Routing. To estimate the number of exchanged messages in the case of covering-based routing we assumed that the probability that a Boolean subscription (or top-k/w subscription proxy) is covered by a previously activated subscription (i.e. proxy) is equal to $cp = 0.5$. Additionally, we assumed that the system is a tree with the average node degree equal to $nd = 3$. For the Boolean system, as shown in Table 3, we estimate $(1-cp) \cdot S \cdot (N-1) + M \cdot log_{nd}(N)$ messages since the system has to be flooded with each non-covered subscription, and since each matching subscription has to be delivered to its subscriber descending down the (ternary) tree. The additional number of $(1-cp) \cdot U \cdot (N-1)$ appears in the case of top-k/w system due the fact that every subscription threshold update also has to be propagated to its proxy subscriptions stored at other nodes in the system. Therefore, we get a high percent increase of 154% in the case of top-k/w subscriptions.

Rendezvous Routing. In the case of Boolean system with rendezvous routing we get $S \cdot log_{nd}(N) + P \cdot log_{nd}(N) + M$ messages since every activated subscription and published publication is delivered to the rendezvous node, while each matching publication is delivered to its subscriber. In the case of the top-k/w system, an additional network cost can be present in the case when neighboring nodes are informed about subscription threshold

Table 3: Number of exchanged messages for different routing strategies and types of subscriptions.

Routing strategy	Subscription type	Number of exchanged messages per type				Total number of exchanged messages	Percent increase
		Subscribing	Publishing	Delivering	Updating		
Publication flooding	Boolean		$(N-1)\cdot P$			$255\cdot 10^6$	0%
	Top-k/w		$(N-1)\cdot P$			$255\cdot 10^6$	
Subscription flooding	Boolean	$S\cdot(N-1)$		M		$0.26\cdot 10^6$	996%
	Top-k/w	$S\cdot(N-1)$		M	$U\cdot(N-1)$	$2.81\cdot 10^6$	
Covering-based routing	Boolean	$(1-cp)\cdot S\cdot(N-1)$		$M\cdot log_{nd}(N)$		$0.83\cdot 10^6$	154%
	Top-k/w	$(1-cp)\cdot S\cdot(N-1)$		$M\cdot log_{nd}(N)$	$(1-cp)\cdot U\cdot(N-1)$	$2.10\cdot 10^6$	
Rendezvous routing	Boolean	$S\cdot log_{nd}(N)$	$P\cdot log_{nd}(N)$	M		$5.20\cdot 10^6$	0.58%
	Top-k/w	$S\cdot log_{nd}(N)$	$P\cdot log_{nd}(N)$	M	$nd\cdot U$	$5.23\cdot 10^6$	
Basic gossiping	Boolean		$sp\cdot N\cdot P$			$25.60\cdot 10^6$	0%
	Top-k/w		$sp\cdot N\cdot P$			$25.60\cdot 10^6$	
Informed gossiping	Boolean	$nd\cdot S$	$sp\cdot N\cdot P$			$25.60\cdot 10^6$	0.12%
	Top-k/w	$nd\cdot S$	$sp\cdot N\cdot P$		$nd\cdot U$	$25.63\cdot 10^6$	

changes. In this analysis we assume that only direct neighbors are informed about threshold changes. Therefore, we get $nd\cdot U$ messages related to subscription updating, which results in a low percent increase of 0.58%.

Basic Gossiping. To estimate the number of exchanged messages in the case of gossiping strategies we assumed that the probability of publication spreading is equal to $sp = 0.1$. Therefore, for both types of subscriptions we get the same total number of exchanged messages $sp\cdot N\cdot P$.

Informed Gossiping. In case of informed gossiping, we additionally assume that each direct neighbor of a subscriber node is informed about a subscription which then, when compared to basic gossiping, results in additional $nd\cdot S$ messages in the case of Boolean subscriptions and $nd\cdot S + nd\cdot U$ in the case of top-k/w subscriptions. This gives a low percent increase of 0.12% for top-k/w subscriptions.

To conclude, the evaluation confirms the results of our previous analysis in Section 4 and identifies four routing techniques which do not cause significant messaging overhead compared to Boolean implementation: publication flooding, rendezvous routing, basic gossiping, and informed gossiping. However, if we look at the number of exchanged messages, one can see that both subscription flooding and covering-based routing generate a small number of messages compared to publication flooding and gossiping. This is due to our scenario setup which generates a large number of publications, while subscriptions are rather static. Therefore, one should not discard subscription flooding, and especially covering-based routing as non-viable routing techniques for distributed top-k/w publish/subscribe systems since the number of generated messages may be comparable, or even lower than the number of exchanged messages in systems with rendezvous routing.

6. Related Work

The idea to rank publications in publish/subscribe systems according to a subscription has been developed and published in parallel in our paper [10] and the following two papers [11, 12]. However, the other authors did not recognize the importance of sliding-window in publish/subscribe setting. In [12] the authors present an approach in which a static time of expiration is associated with each subscription in the system, and afterwards, the same group of authors introduced two interesting concepts in publish/subscribe systems: publication freshness [24], and subscription novelty [25]. Publication freshness is modeled by linear decrease of publication scores in time, while subscription novelty is described by increasing scores of incoming publications for subscriptions that have rarely been satisfied in the past. Additionally, the same group of authors presented PerfSIENA [26], an implementation of their ranking mechanism in SIENA [27] based on user preference. In the latter paper, the authors finally consider the top-k/w matching model, but do not focus on its distributed implementation, which is the main contribution of this paper. Similarly, the top-k/w matching model is examined in [28], but the authors focus mainly on issues related to quality of service.

6.1. Data Stream Processing Systems

The processing of continuous sliding-window top-k queries (top-k/w processing) over data streams has attracted considerable attention in recent years due to its potential application in many different areas such as environmental monitoring using wireless sensor networks, information filtering, computer and telephone network monitoring, financial and stock trade monitoring, etc. Existing solution on top-k/w processing ([29, 8, 9, 6, 30, 10, 31, 7, 20, 19, 21]) assume centralized processing at a single network node and thus differ significantly from the distributed top-k/w processing approach we present in this paper. They can be classified in two categories: deterministic approaches ([8, 9, 6, 30, 20, 19, 21]) which produce correct results to defined queries, and probabilistic approaches ([29, 7, 19]) which generate errors and thus produce approximate results, but are in general more efficient and require less memory than the deterministic approaches. Furthermore, as a top-k/w query continuously identifies k best-ranked data objects

in the query window with respect to an arbitrary scoring function, we can additionally classify existing algorithms according to the type of supported scoring functions. Examples are distance [29, 9, 6], aggregation [8, 30, 31] and relevance [20, 21] scoring functions.

6.2. Distributed Publish/Subscribe Systems

Related work on distributed publish/subscribed systems mainly deals with efficient routing strategies, or introduces novel subscription languages and matching function. Publication flooding is easily implemented as a routing strategy and is thus used in the first publish/subscribe prototypes, e.g. [32]. However, this strategy is rarely used since it does not scale in terms of communication overhead. It mainly serves as a referent routing strategy in experimental evaluations, e.g. [33] and JEDI [34], but is sometimes also used in implementations, e.g. Gryphoon [35].

Subscription flooding, as the opposite routing strategy to publication flooding, is preferable in scenarios with high publication rates when subscriptions rarely change. However, it generates high communication overhead when subscriptions change at a high rate [36, 37]. Since subscribers can be reached in a single hop while non-matching publications can be filtered at the publisher side, some recent distributed publish/subscribe systems apply this strategy, e.g. RUBDES [38] and MEDYM [39].

Selective routing strategies (i.e. covering-based and rendezvous routing) are the most popular routing strategies in distributed publish/subscribe systems. Since they are deterministic and do not flood the overlay network with messages, they are much more interesting for the research community than gossiping and flooding strategies. Covering-based routing is first introduced in SIENA[40], and has been applied in many distributed publish/subscribe systems such as JEDI [34], REBECA [37] and PADRES [41]. This routing strategy has attracted a lot of attention of the research community which has resulted in many different improvements of this routing strategy, e.g. [18, 42, 43].

Rendezvous routing is proposed in Scribe [44] and is used in many other distributed publish/subscribe systems such as Bayeux [45], Hermes [46] and Meghdoot [47].

Gossiping routing strategies (i.e. basic and informed gossiping) are also very popular within the publish/subscribe research community since they lead to high reliability, robustness and self stabilization [23]. Examples of distributed publish/subscribe systems which employ these routing strategies are Costa at al. [48], SpiderCast [49] and Tera [50]. Costa at al. [48] present a content-based publish/subscribe system with three different approaches to the basic gossiping, while other works employ informed gossiping as a routing strategy.

All of the previously mentioned works are relevant to our work, however, we present a novel publish/subscribe matching model and adapt existing routing strategies to it. The top-k/w publish/subscribe model was initially introduced in [10], however, it does not consider distribution-related issues nor provides experimental evaluation of its performance, but mainly focuses on centralized processing with an original probabilistic algorithm.

We have designed the first distributed solution for k-NN sliding window computation [13] which is based on the general model and ideas presented in this paper. However, the specific solutions are adjusted to processing k-NN queries in Euclidean spaces over the CAN peer-to-peer network. Furthermore, our distributed k-NN top-k/w solution uses regular grid as an indexing structure, rendezvous routing, and specific protocols for updating query indexing thresholds between the processing nodes.

In the area of distributed publish/subscribe systems, the most relevant work to ours is [22] which introduces a new type of stateful subscription called the parametrized subscription. We can regard a top-k/w subscription proxy as parametrized subscriptions with subscription threshold as the only parameter. The authors discuss filtering-based routing, which is a simpler version of covering-based routing, both in a centralized and distributed architecture. Every parameter change is published as a publication on the corresponding parameter topic. The authors conclude that such changes have to be propagated to all network nodes, as we have also concluded for covering-based routing. Please note that the results of our analysis for various routing strategies can easily be applied to publish/subscribe systems supporting such parametrized subscriptions.

7. Conclusion

In traditional publish/subscribe systems, subscription is a stateless Boolean function for which a decision whether to deliver a publication to a subscriber is made based only on the publication and subscription content, and does not take into account any additional information present in the system. In this paper we show that this approach results in an unpredictable number of matching publications which may cause user dissatisfaction with a provided service by delivery of either too many or too few publications.

This paper presents the top-k/w matching model, a novel publish/subscribe matching model which enables a subscriber to control the number of publications it receives per subscription within a predefined time period. In this model, each subscription defines a time-independent scoring function and parameters k and window-size w such that, at a point in time t, parameter k restricts the number of delivered publications to the k best scored publications that are published between $t - w$ and t. Additionally, we give a formal proof that the Boolean matching model is a special case of the top-k/w matching model. Furthermore, the paper gives examples of distance, aggregation and relevance scoring functions which are supported by the top-k/w publish/subscribe model.

On the foundations of centralized top-k/w processing algorithms, we analyze possible top-k/w model implementations in distributed environments. For each of the commonly used routing strategies in distributed environments we explain how it can be adapted to support the top-k/w matching model. Among them, we also identify the ones which are particularly well suited for large-scale top-k/w publish/subscribe systems.

Using the case study with k-NN subscriptions and both real and synthetic data sets, we experimentally evaluate and compare the top-k/w and Boolean matching models and estimate the number of exchanged messages for different routing strategies in distributed Boolean and top-k/w systems. Based on the results of our evaluation, we conclude two things. First, the top-k/w matching model is indeed capable to control the number of matching publications per subscription, which, together with built-in support for flexible subscriptions, represents a significant enhancement of publish/subscribe systems. Second, the top-k/w matching model can be efficiently implemented in distributed environments using the identified routing strategies. An example implementation for continuous sliding-window k-NN processing over a CAN overlay network is available in [13].

References

[1] M. Balazinska, A. Deshpande, M. J. Franklin, P. B. Gibbons, J. Gray, M. Hansen, M. Liebhold, S. Nath, A. Szalay, V. Tao, Data management in the worldwide sensor web, IEEE Pervas. Comput. 6 (2007) 30–40.

[2] R. K. Ganti, F. Ye, H. Lei, Mobile crowdsensing: current state and future challenges, IEEE Communications Magazine 49 (11) (2011) 32–39.

[3] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2003) 114–131.

[4] Y. Zhou, K.-L. Tan, F. Yu, Leveraging distributed publish/subscribe systems for scalable stream query processing, LNCS 4365 (2007) 20–33.

[5] G. Mühl, L. Fiege, A. P. Buchmann, Filter similarities in content-based publish/subscribe systems, in: ARCS, 2002.

[6] K. Mouratidis, D. Papadias, Continuous nearest neighbor queries over sliding windows, IEEE Trans. on Knowl. and Data Eng. 19 (2007) 789–803.

[7] C. Jin, K. Yi, L. Chen, J. Yu, X. Lin, Sliding-window top-k queries on uncertain streams, The VLDB Journal 19 (2010) 411–435.

[8] K. Mouratidis, S. Bakiras, D. Papadias, Continuous monitoring of top-k queries over sliding windows, in: SIGMOD, 2006.

[9] C. Böhm, B. C. Ooi, C. Plant, Y. Yan, Efficiently processing continuous k-nn queries on data streams, in: ICDE, 2007.

[10] K. Pripužić, I. Podnar Žarko, K. Aberer, Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w, in: DEBS, 2008, pp. 127–138.

[11] A. Machanavajjhala, E. Vee, M. Garofalakis, J. Shanmugasundaram, Scalable ranked publish/subscribe, in: VLDB, 2008.

[12] M. Drosou, E. Pitoura, K. Stefanidis, Preferential publish/subscribe, in: PersDB, 2008.

[13] K. Pripužić, I. Podnar Žarko, K. Aberer, Distributed processing of continuous sliding-window k-nn queries for data stream filtering., World Wide Web 14 (5-6) (2011) 465–494.

[14] L. Lamport, Proving the correctness of multiprocess programs., IEEE Trans. Software Eng. 3 (1977) 125–143.

[15] Z. M. Edward Chang, A. Pnueli, Logic and Algebra of Specifications, Springer-Verlag, 1991, Ch. The Safety-Progress Classification.

[16] I. F. Ilyas, G. Beskales, M. A. Soliman, A survey of top-k query processing techniques in relational database systems, ACM Comput. Surv. 40 (2008) 1–58.

[17] S. Tarkoma, J. Kangasharju, Optimizing content-based routers: posets and forests, Distributed Computing 19 (2006) 62–77.

[18] A. M. Ouksel, O. Jurca, I. Podnar, K. Aberer, Efficient probabilistic subsumption checking for content-based publish/subscribe systems, LNCS 4290 (2006) 121–140.

[19] K. Pripužić, Top-k publish/subscribe matching model based on sliding window, Ph.D. thesis, University of Zagreb (2010).

[20] K. Mouratidis, H. Pang, An incremental threshold method for continuous text search queries, in: ICDE, 2009.

[21] P. Haghani, S. Michel, K. Aberer, The gist of everything new: personalized top-k processing over web 2.0 streams, in: CIKM, 2010, pp. 489–498.

[22] Y. Huang, H. Garcia-Molina, Parameterized subscriptions in publish/subscribe systems, Data Knowl. Eng. 60 (2007) 435–450.

[23] R. Baldoni, L. Querzoni, S. Tarkoma, A. Virgillito, Distributed event routing in publish/subscribe systems, in: MiNEMA, 2009.

[24] M. Drosou, Ranked publish/subscribe delivery, in: DEBS PhD, 2009.

[25] K. S. D. Souravlias, M. Drosou, E. Pitoura, On novelty in publish/subscribe delivery, in: DBRank, 2010.

[26] M. Drosou, K. Stefanidis, E. Pitoura, Preference-aware publish/subscribe delivery with diversity, in: DEBS, 2009.

[27] A. Carzaniga, D. S. Rosenblum, A. L. Wolf, Achieving scalability and expressiveness in an internet-scale event notification service, in: PODC, 2000.

[28] X. Lu, X. Li, T. Yang, Z. Liao, W. Liu, H. Wang, RRPS: A ranked real-time publish/subscribe using adaptive QoS, LNCS 5593 (2009) 835–850.

[29] N. Koudas, B. C. Ooi, K.-L. Tan, R. Zhang, Approximate nn queries on streams with guaranteed error/performance bounds, in: VLDB, 2004.

[30] G. Das, D. Gunopulos, N. Koudas, N. Sarkas, Ad-hoc top-k query answering for data streams, in: VLDB, 2007.

[31] Y. Zhang, Computing order statistics over data streams, Ph.D. thesis, University of New South Wales (2008).

[32] B. Oki, M. Pfluegl, A. Siegel, D. Skeen, The information bus: an architecture for extensible distributed systems, SIGOPS Oper. Syst. Rev. 27 (1993) 58–68.

[33] A. Carzaniga, Architectures for an event notification service scalable to wide-area networks, Ph.D. thesis, Politecnico di Milano (1998).

[34] G. Cugola, E. D. Nitto, A. Fuggetta, The jedi event-based infrastructure and its application to the development of the opss wfms, IEEE T. Software. Eng. 27 (2001) 827–850.

[35] G. Banavar, T. Chandra, B. M. andJay Nagarajarao, R. E. Strom, D. C. Sturman, An efficient multicast protocol for content-based publish-subscribe systems, in: ICDCS, 1999.

[36] A. Carzaniga, D. S. Rosenblum, A. L. Wolf, Design and evaluation of a wide-area event notification service, ACM Trans. Comput. Syst. 19 (2001) 332–383.

[37] G. Mühl, Large-Scale Content-Based publish/subscribe systems, Ph.D. thesis, Darmstadt University of Technology (2002).

[38] O. K. Sahingoz, N. Erdogan, RUBDES: A rule based distributed event system, LNCS 2869 (2003) 284–291.

[39] F. Cao, J. P. Singh, Medym: Match-early and dynamic multicast for content-based publish-subscribe service networks, in: ICDCSW, 2005.

[40] A. Carzaniga, M. J. Rutherford, A. L. Wolf, A routing scheme for content-based networking, in: INFOCOM, 2004.

[41] G. Li, H.-A. Jacobsen, Composite subscriptions in content-based publish/subscribe systems, LNCS 3790 (2005) 249–269.

[42] R. Baldoni, R. Beraldi, L. Querzoni, A. Virgillito, Efficient publish/subscribe through a self-organizing broker overlay and its application to siena, Comput. J. 50 (2007) 444–459.

[43] G. Li, V. Muthusamy, H.-A. Jacobsen, Adaptive content-based routing in general overlay topologies, LNCS 5346 (2008) 1–21.

[44] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, P. Druschel, Scribe: The design of a large-scale event notification infrastructure, in: NGC, 2001.

[45] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, J. D. Kubiatowicz, Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination, in: NOSSDAV, 2001.

[46] P. R. Pietzuch, J. M. Bacon, Hermes: A distributed event-based middleware architecture, in: ICDCSW, 2002.

[47] A. Gupta, O. D. Sahin, D. Agrawal, A. E. Abbadi, Meghdoot: content-based publish/subscribe over p2p networks, LNCS 3231 (2004) 254–273.

[48] P. Costa, M. Migliavacca, G. P. Picco, G. Cugola, Epidemic algorithms for reliable content-based publish-subscribe: An evaluation, in: ICDCS, 2004.

[49] G. Chockler, R. Melamed, Y. Tock, R. Vitenberg, Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication, in: DEBS, 2007.

[50] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, S. Tucci-Piergiovanni, Tera: topic-based event routing for peer-to-peer architectures, in: DEBS, 2007.