Ranked Publish/Subscribe Delivery

Extended abstract for DEBS PhD Workshop

Marina Drosou
Supervisor: Evaggelia Pitoura
Department of Computer Science
University of Ioannina, Greece
mdrosou@cs.uoi.gr

ABSTRACT

Publish/subscribe systems offer an attractive alternative to classic search methods by allowing users to specify their interests once and be notified whenever new interesting information becomes available. All relevant information for a user is considered equally important. This can result in large amounts of notifications being delivered to users. In this PhD thesis, we consider the problem of ranking the notifications reaching the users based on a number of criteria, such as relevance and diversity, to control the number of delivered events and increase user satisfaction.

1. INTRODUCTION

In today's world, there is a great amount of information available for every user. However, locating the most valuable or important information can prove out to be an overwhelming task, due to the great volume of accessible data. Publish/subscribe systems aim at tackling this problem by disburdening the users of the need to repeatedly search for new information. A proactive model is used instead, where users specify their interests via subscriptions and the system automatically forwards all new interesting published information to them.

This PhD thesis aims at increasing user satisfaction by the received results by proposing methods that will control the number of results delivered to the users, mainly through ranking them according to their importance.

We consider that the importance of results can be described by three factors: (i) their *relevance* to user interests, (ii) their *diversity* and (iii) their *freshness*. Relevance is important so that users are only notified about the most interesting events, while diversity ensures that the received notifications are not referring to the same or similar events. Finally, freshness is also important in the scope of publish/subscribe systems, where the flow of information is continuous. These three factors can be studied independently from each other, since their application is orthogonal. However, this thesis aims at designing and implementing techniques that will allow the combined, efficient application of all three factors for the final ranking of notifications and their forwarding to the users.

Ranked delivery in publish/subscribe systems is a new, interesting research area. The problem had not been addressed until recently. In [14], ranking in publish/subscribe environments is also considered. However, it is viewed in a different way. In a sense, the reverse problem is considered. Instead of locating the most relevant events to each subscription, the authors aim at recovering the most relevant matching subscriptions to a published event. To quickly recover the top subscriptions related to an event, scored indices are built over the subscriptions. Another work that also deals with the problem of ranked publish/subscribe is [17]. In the proposed model, a subscriber receives the k most relevant events per subscription within a window w which can be either time-based or event-based. For each user subscription, a queue is maintained. This queue buffers those events that are relevant to the subscription and have a high probability to enter the top-k results at some point in the future. The focus is on efficiently maintaining this buffer queue, while in our work so far, we aim at specifying and computing the importance of each new event. Our initial results on combining relevance and diversity along with freshness have been presented in [8, 9]. In [8], we introduced preferential subscriptions, presented ways to compute the importance of events and discussed forwarding issues. In [9], we formalized our model, including diversity, and presented a number of delivery modes for forwarding events to users. Here, we summarize our current work and elaborate on our further ideas.

Ranked delivery can be used with a wide variety of applications. Due to the tremendous volume of information propagated through them, we believe that social systems, such as Twitter [4] or Facebook [1], can benefit from it. In such systems, users receive a great amount of real-time notifications about various topics of interest for which they have subscribed. Such topics may include, for example, actions of other specific users, or "friends". Today, all notifications of interest are considered equally important and are presented to the user arbitrarily. However, users are probably more interested in some of their friends or some specific kinds of events. Ranked delivery will enable users to receive ordered information so that they do not have to scan all of the available notifications to locate the ones that are more interesting to them. Also, diversity will ensure that users do not receive the same information from many different sources over and over again, something that is quite common in such systems

DEBS PhD Workshop 2009

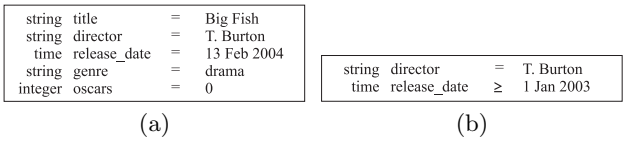

Figure 1: (a) A notification and (b) a matching subscription.

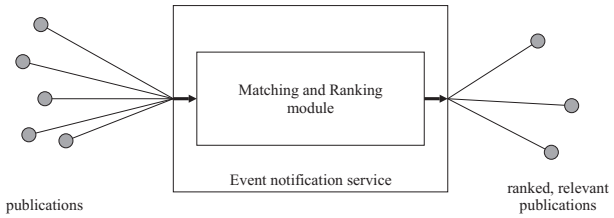

Figure 2: Basic ranked publish/subscribe system.

today.

In the rest of this paper, we present research challenges in the area of ranked publish/subscribe delivery. We both describe our results so far and also emphasize other alternative solutions to highlight the broadness of the problem. The remaining of this paper is structured as follows. In Section 2, we summarize our model. In Sections 3, 4 and 5, we discuss how diversity, preferences and freshness, respectively, influence the importance of events, while in Section 6, we elaborate on how we can combine those criteria to rank events. In Section 7, we present our current system design and discuss improvements. Finally, Section 8 summarizes our work and future plans.

2. MODEL OVERVIEW

In the rest of this paper, we assume a content-based publish/subscribe system, since such systems offer greater expressiveness to the subscribers than the topic-based alternative. Our techniques, however, can be adapted to topic-based systems as well. Generally, in content-based systems, matching depends on the actual content of notifications, while in topic-based ones, it depends on some classification of notifications into predefined topics. For a detailed categorization of publish/subscribe systems, the reader is referred to [12].

We consider event notifications as sets of typed attributes, capturing the content of each event, similar to [7, 13]. Each notification consists of an arbitrary number of attributes that take values according to their type. Subscriptions on the other hand consist of a set of constraints on the values of specific attributes. Such constraints are enforced via the use of binary operators, such as $=$, $\neq$, $<$, $>$, $\leq$, $\geq$, $substring$, $prefix$ and $suffix$. A notification *matches* a subscription (or a subscription is *covered* by a notification), if and only if, the notification satisfies all of the subscription's attribute constraints (Figure 1).

Since the flow of information is continuous, it is important to define which events should be compared with each other in order to yield the most important of them, where "important" may mean diverse, highly relevant, fresh or a combination thereof. In general, each event e of a publish/subscribe system (and the corresponding notification) is associated with a number of time instants:

1. The time e is published ($tpub_e$)
2. The time e reaches the event-notification service ($tserv_e$)
3. The time e is matched against subscriptions ($tmatch_e$)
4. The time e is forwarded to the user ($tforw_e$) and
5. The time e is actually received by the user ($trecv_e$)

Due to the asynchronous mode of communication offered by publish/subscribe systems, arbitrary delays may be inserted between those time instants. Those delays are also generally different for each event. This can affect all ranking methods.

Also, in contrast to traditional publish/subscribe systems which are stateless, events cannot be disregarded after their matching and delivery to the users, since new events have to be compared against them in order to be ranked. However, keeping all past events in the system is neither practical nor particularly useful, since in practice, we are not interested in locating the top-ranked events ever but rather the more recent (fresh) top-ranked events, since old top-ranked events (should) have already been delivered to the user in the past.

Therefore, in this thesis, we investigate time-partitioning methods. Such methods can be either (i) *periodic* or (ii) *window-based*. In periodic methods, time is divided into periods and matching events are buffered until the end of the period. When the period ends, top-ranked events are forwarded to the users. Though this is simple to be done on a per-user basis, various practical issues arise in a distributed, multi-user environment, where the matching events are different for each user and also different users may wish periods of different lengths. A possible solution is to cluster users according to the similarity of their preferences or subscriptions and buffer matching events per cluster. This would greatly decrease the maintenance cost of the system but would result in approximate solutions of the problem.

Besides periodic techniques, window-based ones can also be used. Actually, computing top-ranked events over sliding windows seems more intuitive to the problem due to the continuous flow of new information. Such windows can be either time-based or event-based. Clustering can also be applied in this case as well.

In both cases, published events and notifications must be buffered for some time prior to their final dismissal from the system. Generally, a ranked published subscribe system will include an enhanced matching mechanism that, besides locating all matching subscriptions, will also be able to compute the importance of each event and rank the incoming notifications according to the selected criteria (Figure 2).

3. DIVERSITY

A natural consequence of all traditional ranking methods is that, often, the top-results are very similar to each other. This happens because, according to the ranking criterion used each time, data items, or notifications in our case, containing the same, highly important piece of information are all ranked in the top positions, even if they are redundant. For example, when in Greece, Google's top-10 results for the keyword query "uoi" are all about the University of Ioannina, while most of Google Video's top-10 results are about the University of Illinois.

However, besides pure accuracy achieved by the ranking criterion, there are also other factors that can increase user satisfaction, such as retrieving results on a broader variety

of topics, i.e. increasing the diversity of results.

There are two different perspectives on achieving diversity:

1. Avoiding overlap among results, i.e. choosing data items that are dissimilar to each other.
2. Increasing coverage, i.e. choosing data items that cover as many different topics as possible.

Most research work so far in the literature views the problem using the first perspective mentioned above and various measuring alternatives have been proposed, all based on the notion of similarity (or distance) among the selected data items. For example, [11] aims at maximizing the minimum distance among the selected data items, while [19] attempts to minimize their average similarity. Independently of how one wishes to view diversity, we define the diversification problem as follows:

Definition 1. The DIVERSIFY(k) problem is to select the k out of all results that exhibit the maximum diversity.

Generally, the problem of choosing diverse results has been proved to be NP-hard [10]. This follows from the Max-Coverage and Set-Cover problems. Intuitively, to find the most diverse subset S^* of all results M, all possible combinations of k out of $|M|$ items have to be computed in order to select the one with the maximum diversity. Note that, the term "set" is used loosely here, as the order that the results of S^* will be presented to the user is also important.

Due to the complexity of the problem, which makes impractical the computation of optimal solutions, heuristics are used to locate approximate ones [11]. The most common approaches are (i) the greedy heuristics, where items are selected or disqualified one by one until k of them remain and (ii) the interchange heuristics, where a random solution is first selected and then, we proceed by interchanging selected and disqualified items so that the resulting solution is improved.

In our work so far [9], we have used a greedy heuristic and showed that the approximate solutions located by it were very close to the optimal ones. In the future, we also plan to experiment with the use of interchange heuristics, since it seems that they also suit continuous flows of information. In such environments, we could check if a new data item can be interchanged with an older one to improve the overall diversity of selected items.

Generally, a research challenge in the context of publish/subscribe systems is that the computation of diversity is continuous. In addition to this, we have seen that a major difficulty for efficiently computing good solutions is that the $k-1$ most diverse items of M are not necessarily a subset of its k most diverse items. For example, consider as data items the points on the circumstance of a circle and their euclidean distances. The two furthest apart points are (any) two antidiametric ones. However, no antidiametric points can be found among the three most diverse points (Figure 3).

We are also interested in the second perspective for viewing diversity, i.e. covering. In particular, we would like to cover as many subscriptions as possible. Our main motivation is that, since users take the time to specify a number of subscriptions, they are most probably interested in receiving notifications matching all of them. There are two research directions for this issue. First, we could control the amount of events that are delivered to each user due to each subscription. This means that all user subscriptions should be responsible for a number of events forwarded to the user, so that users are not overwhelmed by (similar) events matching just a few of their subscriptions. This can be achieved by associating a delivery rate with each subscription. This rate should depend on how diverse the subscription is from the others. For example, in a group of similar subscriptions, each one of them should contribute less events to the user's results than a single subscription that is very different from all the others. The similarity between two subscriptions could be measured based on the number of their common attributes or even on the expected diversity of the events that will match them. Second, we could select for delivery these notifications that match as many subscriptions as possible. This would ensure that all user subscriptions are covered by the delivered events, while at the same time, the amount of delivered notifications would be greatly reduced.

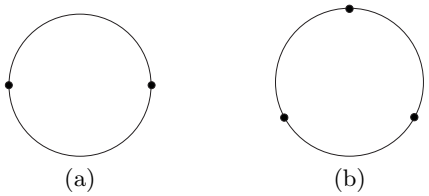

Figure 3: Diverse points on a circle.

4. PREFERENCES

We assume that preferences are associated with user subscriptions. Users specify subscriptions to express their interests. Subscriptions are used to filter out irrelevant notifications and select only the notifications about events that are interesting to the user. So far, in our work, we have assumed preferences among subscriptions. User-defined preferences among those subscriptions are employed to rank the notifications and deliver to the user only the most preferable ones. We first review our results in Section 4.1 and then explore the alternative of defining preferences among specific attribute values in Section 4.2.

4.1 Subscription-based preferences

User preferences are expressed by associating each subscription with a degree of interest, called *preference rank*. These preference ranks can be defined using either a quantitative or a qualitative preference model. In the quantitative case, preference ranks are defined explicitly by users via the use of numeric scores, usually normalized in $[0, 1]$ (e.g. "I prefer drama films with a score of 0.6"). In the qualitative case, users define binary preference relations among their subscriptions (e.g. "I prefer drama films more than horror ones"). In the latter case, the preference ranks of those subscriptions are extracted using the multiple level winnow operator as described in [9]. Generally, assuming that binary relations follow a strict partial order, subscriptions are organized in a directed graph according to those relations. Then, subscriptions are recovered in levels using a topological sort algorithm, with the most preferable subscriptions recovered first. Each subscription is given a preference rank according to its level l. To do this, a monotonically decreasing function of l is used.

A subscription associated with a preference rank is called a *preferential subscription*:

Definition 2. A preferential subscription ps_i^X of user X is a pair of the form $ps_i^X = (s_i, prefrank_i^X)$, where s_i is a subscription and $prefrank_i^X$ is a real number in $[0, 1]$ that expresses the degree of interest of X for s_i.

Each user specifies preferences using one of the above models. How extracted preference ranks can be compared with explicit ones in the case both models are used by a specific user simultaneously is an interesting open issue.

Let P^X be the set of preferential subscriptions (or *profile*) of user X. These subscriptions can be used to rank the published notifications and deliver to the user only the highest ranked ones. To do this, we compute the rank of each published event for each user. The highest this rank, the more interesting the event is to the user. An event's rank for a user is computed based on the preference ranks of the subscriptions in P^X that cover it:

Definition 3. Given an event e, a user X, the set P^X of the user's preferential subscriptions and the set $P_e^X = \{(s_1, prefrank_1^X), \ldots, (s_m, prefrank_m^X)\}$, $P_e^X \subseteq P^X$, such that s_i, $1 \leq i \leq m$, are the subscriptions that e matches, the event rank of e for X is equal to $rank(e, X) = \mathcal{F}(prefrank_1^X, \ldots, prefrank_m^X)$, where $\mathcal{F}$ is a monotonically increasing function.

Instead of using all matching subscriptions to compute event ranks we can also use only the most specific of them, since these subscriptions express better the user's degree of interest for the events.

4.2 Attribute-based preferences

The method described above for expressing preferences requires users to define preferences among their subscriptions as a whole. An alternative way for users to express their interests is to specify preferences by defining binary preference relations among specific attribute values. Then, an aggregate rank can be computed for each subscription based on those relations. Both a qualitative as well as a quantitative approach can be followed as before. For example, a user may state the following attribute-based, qualitative preferences:

- actor = R. De Niro $\succ$ actor = H. Ford
- actor = R. De Niro $\succ$ actor = A. Paccino
- genre = comedy $\succ$ genre = drama

where the symbol "$\succ$" denotes preference of the first item over the second one. Preferences for each attribute should again follow a strict partial order. The preferences of user X for each attribute are organized in a directed graph. Then, the multiple level winnow operator is employed. Attribute values appearing in higher winnow levels are more preferable to the user. For each attribute value a preference rank for a specific user is computed as a function of its winnow level for that user. As in the previous model, a monotonically decreasing function function should be used. If an attribute value does not appear in any of the graphs then its preference rank for X can be considered to be equal to 0.

Using this model, event ranks can be computed based on all of the corresponding notification's attributes. Each attribute will satisfy the user by a certain degree of interest, as this is specified by the user's attribute-based preferences. Event ranks can be computed as the normalized sum or average of those degrees of interest. In this way, we can allow a finer granularity for expressing user preferences. For example, based on the above preferences, a comedy with R. De Niro will be ranked higher than a drama with the same actor.

4.3 Ranking based on preferences

Given the computed event ranks, event notifications can be ranked and delivered according to them. In our work so far, we have used a top-k delivery model for this. Alternatively, instead of delivering to users the k highest ranked events, we could consider a skyline model where an event is delivered to the user only if it belongs to the user's skyline, i.e. there is no past event for which all attribute values are more preferable to the corresponding attribute values of the new event.

5. FRESHNESS

Another important criterion in publish/subscribe systems, is freshness, i.e. how recent, or fresh, the delivered notifications are, since older notifications are expected to be of less interest to the users.

A notification should be considered more important closer to the time of its publication than later in time. During the elaboration of this thesis, we plan to examine different methods to achieve this, such as using aging techniques. In this way, each event will be associated with an age (how long the event has been in the system). The older an event, the more its importance will decrease. For this purpose, a function $\mathcal{F}_\mathcal{A}$ should be used to weight the importance of each event. This function will monotonically decrease along with time. Assuming a time limit τ, after which an event is not considered important anymore, an example of such a function is:

$$\mathcal{F}_\mathcal{A}(e, t) = \left(1 - \frac{t - tpub_e}{\tau}\right)$$

where e is an event and $t \in [tpub_e, tpub_e + \tau]$. This function degrades linearly but it would be interesting to compare the performance of functions with different properties, e.g. a function that degrades exponentially.

The freshness of a notification depends on the system overhead, i.e. the latency introduced by the underlaying network of communication and also the application of the matching and ranking algorithms. A number of issues arise, such as the fact that notifications may reach different users at different times, not necessarily in the order they were published. This is a general problem due to the distributed nature of the system but can cause non-intuitive effects, such as different users with the same preferences receiving different notifications. We could explicitly view notifications that have spent too much time being propagated in the network as old and block them from being forwarded to users, since newer notifications have probably already been forwarded to them.

Finally, an interesting direction for research is that of updatable events. We call an event updatable when newer versions of it can possibly be published in the future. For example, in the case of a sensor network where sensor units periodically publish new measurements, published events do not contain new information per se but rather an update of the older measurements (and the corresponding events). In this case, there is no reason for the old notification to continue spreading through the network or to be considered more important than the new one due to user preferences. Other open issues when update semantics are employed con-

cern the ordering of events. For example, should a new even replace an older version of itself in the various buffers throughout the system or be appended to the end?

6. COMBINING CRITERIA

Diversity, relevance and freshness are all important factors for ranking event notifications in a publish/subscribe system. There are various options for combining them into a single ranking method.

First, let us assume an event e, a user X and its event rank $rank(e, X)$ according to X's preferences. We can combine the relevance and freshness factors as follows. As explained in Section 5, we consider that the importance of published events degrades along with time, so we can say that e's importance at time t, where $t \in [tpub_e, tpub_e + \tau]$, is equal to:

$$frank(e, X, t) = \mathcal{F}_{\mathcal{A}}(e, t) rank(e, X)$$

However, the final results must also be diverse with each other. Assuming than we have computed the $frank$s of all the available notifications, we must select the k most diverse of them. A basic way to combine those measures is to follow a linear approach. This was first proposed in [6] and is also adopted in our work in [9]. Following this approach, a diversification factor σ is chosen, $\sigma \in [0, 1]$, and the objective function we wish to maximize is of the form

$$\sigma \cdot \frac{\sum_i frank(e_i, X, t)}{|S|} + (1 - \sigma) \cdot div(S)$$

where $div(S)$ is the exhibited diversity of the set of chosen data items S.

Due to the diversity requirement, this problem is also NP-hard. Therefore, we have to design heuristics to reduce the high complexity of the problem. A first, greedy approach, similar to the one used in [19], is to first choose the two most preferable and diverse data items and then proceed by adding to the solution, one by one, data items that maximize the above function. The main challenge of the problem remains that diversity is a measure that can not be computed for each data item independently but is rather a quantity that characterizes a set of data items as a whole.

The problem (minus the freshness factor) has also been viewed as an optimization problem in [18], albeit in a more mathematical approach, where approximate solutions are located via relaxation and optimization methods.

Following the covering view of the diversity problem, another option would be to choose k important, fresh events that cover as many user subscriptions as possible.

7. DiveR: A PUBLISH/SUBSCRIBE SYSTEM WITH RANKING FUNCTIONALITY

In this section, we report on our current design and implementation of DiveR, a distributed, diversity-aware publish/subscribe system with ranking functionality. In our system design, we have chosen to decouple diversity and preferential ranking computations by using different system modules to treat them.

Generally, measuring diversity is a complex issue, since new events are published through various sources spread over the network. A first approach would be to collect all published events in a single site and perform diversification there. However, this solution would clearly not scale well due to the very large amount of users and generated events. Thus, distributed architectures are more suitable for this problem. We consider that a two-level, super-node based architecture will be better able to support a distributed solution for diverse notification delivery.

The initial, high-level architecture of DiveR is shown in Figure 4. Each user is connected to a super-node of the system. For now, let us assume that users connect to super-nodes randomly. Generally, new generated events of a user are forwarded to the user's super-node. That super-node is responsible for forwarding the events to other super-nodes with interested subscribers connected to them.

Each super-node runs a diversification module to improve the diversity of events propagated through it. There are two options for achieving diversification at super-node level, both of which aim at reducing the number of redundant events, i.e. events that contain similar information. First, each super-node could collect (buffer) recent events published by the nodes connected to it, using some sliding-window or periodic method. Then, it would propagate to the rest of the network not all published events by the users attached to it, but only the most diverse among them.

The second option is to perform diversification not on the events published by a specific group of users but rather on groups of events containing similar information. To achieve this, we assume that user subscriptions are partitioned into groups via some clustering method [15]. After this, each group is assigned to a super-node. A published event is then forwarded to the super-node which is responsible for the group of subscriptions similar to the event. Each super-node is responsible for diversifying the events matching each of its subscription groups. Note that, matching and diversification are performed at the same time since only matching events need to be diversified (the others will be dropped).

None of these two methods will provide optimal solutions to the diversification problem. However, we believe that the approximate solutions provided will improve diversity, as our initial experiments with approximate solutions have shown [9]. We also believe that the second option, i.e. applying diversification per group of subscriptions, will be better, because of the collection of similar events to specific super-nodes of the system. An open topic in this process is the coordination among super-nodes for propagating events and exchanging information. For example, when subscription clustering is applied, super-nodes could summarize the subscriptions assigned to them via structures like bloom-filters [5] or hash-sketches [16] so that a distributed cluster index can be built as shown in Figure 4.

As for preferential ranking, each super-node of the system will be responsible for the ranked delivery of events to each of the subscribers connected to it. Note that, an event's rank has already been computed during the matching/diversification phase. For this reason, the actual content of the events is not needed for the ranking process. Therefore, for top-k delivery the ranking module of each super-node needs to maintain only a list of previously sent event ranks per subscriber. This list can be maintained, for example, using a sliding-window method. Although this is a per user approach, the cost is not expected to be very high because (i) each super-node is responsible for a portion of the system's subscribers and (ii) only a list of k numbers needs to be maintained per subscriber. A possible research direction is the application of information clustering so that

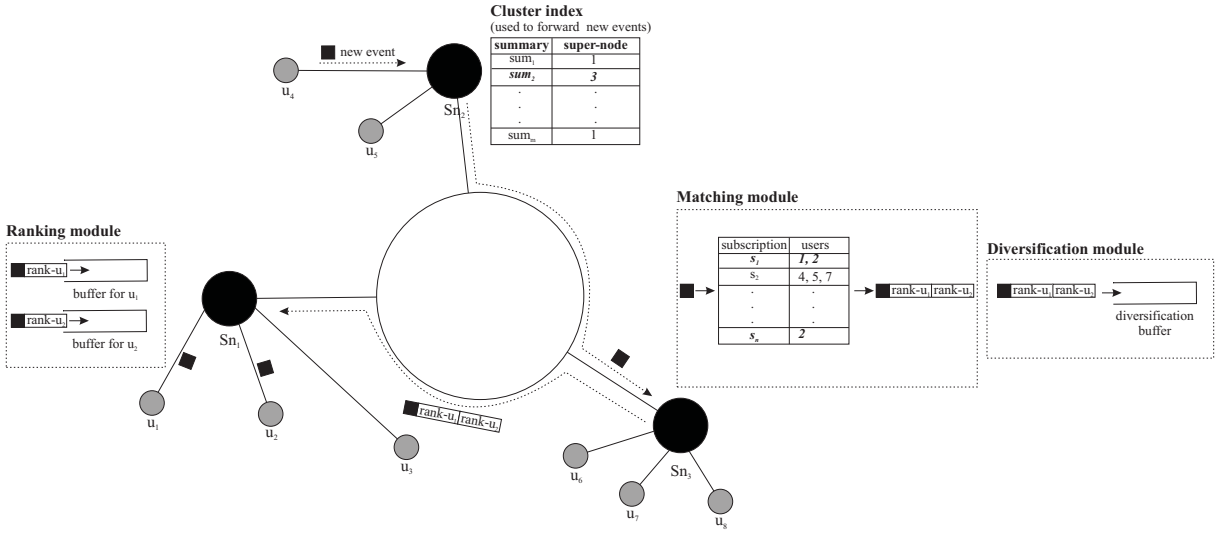

Figure 4: Event forwarding in DiveR.

users who receive similar events are connected to the same super-node. We performed some initial experiments that showed this can improve performance [8].

Our intitial steps towards ranked publish/subscribe delivery have been implemented using the SIENA event notification service [3]. Our prototype, termed PrefSIENA, is available for download [2]. It supports ranked delivery based on user preferences and diversity. The freshness criterion is employed to resolve ties. PrefSIENA also offers three delivery modes: a periodic one and two sliding-window variations. PrefSIENA can be employed in a distributed environment using a hierarchical topology of servers. However it is not optimized for such environments as at the time of its development our research focus remained on exploring ranking methods.

8. SUMMARY

Inspired by the growth of publish/subscribe systems and the convenient model of communication they offer to users, we seek to increase user satisfaction. A strong assumption made by all publish/subscribe systems so far is that all user subscriptions are equally important. While this may be true for a number of applications, such as sensor deployment networks, other applications, such as social networks and recommendations, require the ranking of information that is presented to users. The objective of this PhD thesis is to increase user satisfaction by the received results, thus boosting the acceptability of publish/subscribe systems in the new, exciting area of social information exchange.

We allow users to express their interests through preferences and also account for diversity and freshness when ranking notifications. Our work so far has focused on our model for information ranking and an initial implementation. Our future plans are to develop ranking methods for publish/subscribe systems that can be efficiently applied in a distributed environment.

9. REFERENCES

[1] Facebook. http://www.facebook.com.

[2] PrefSIENA. http://www.cs.uoi.gr/∼mdrosou/PrefSIENA.

[3] SIENA. http://serl.cs.colorado.edu/∼serl/dot/siena.html.

[4] Twitter. http://www.twitter.com.

[5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[6] J. G. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[7] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.

[8] M. Drosou, E. Pitoura, and K. Stefanidis. Preferential publish/subscribe. In *PersDB*, pages 9–16, 2008.

[9] M. Drosou, K. Stefanidis, and E. Pitoura. Preference-aware publish/subscribe delivery with diversity. In *DEBS*, 2009.

[10] E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, May 1990.

[11] E. Erkut, Y. Ülküsal, and O. Yeniçerioglu. A comparison of -dispersion heuristics. *Computers & OR*, 21(10):1103–1113, 1994.

[12] P. T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[13] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe. In *SIGMOD Conference*, pages 115–126, 2001.

[14] A. Machanavajjhala, E. Vee, M. N. Garofalakis, and J. Shanmugasundaram. Scalable ranked publish/subscribe. *PVLDB*, 1(1):451–462, 2008.

[15] T. Milo, T. Zur, and E. Verbin. Boosting topic-based publish-subscribe systems with dynamic clustering. In *SIGMOD Conference*, pages 749–760, 2007.

[16] N. Ntarmos, P. Triantafillou, and G. Weikum. Distributed hash sketches: Scalable, efficient, and accurate cardinality estimation for distributed multisets. *ACM Trans. Comput. Syst.*, 27(1), 2009.

[17] K. Pripuzic, I. P. Zarko, and K. Aberer. Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w. In *DEBS*, pages 127–138, 2008.

[18] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *RecSys*, pages 123–130, 2008.

[19] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.