A-TOPSS – A Publish/Subscribe System Supporting Approximate Matching

Haifeng Liu H.-Arno Jacobsen

Department of Electrical and Computer Engineering and
Department of Computer Science
University of Toronto
10 King's College Road, Toronto, Ontario, Canada, M5S 3G4
hfliu@cs | jacobsen@eecg {.toronto.edu}

1 Overview

The proliferation of pervasive computing devices, the integration of network access technologies (mobile wireless and Internet), and the large amount of information providers offering content are driving the need for an information dissemination model that offers its users highly pertinent information in a demand-driven manner. This can be supported extremely well through the publish/subscribe paradigm, where content providers constitute the publishers of information, while content seekers constitute subscribers.

This paradigm has recently gained great interest in the database community as a solution methodology for information dissemination applications [1, 9, 6] with which alternative models fail to cope adequately.

The publish/subscribe paradigm is a simple to use interaction model that consists of information providers, who publish events to the system, and of information consumers, who subscribe to events of interest within the system. The publish/subscribe system ensures the timely notification of subscribers upon event occurrence. Events can be seen as data items (e.g., tuples, columns, or tables) in the relational database model and subscriptions closely resemble database queries. From this point of view, publish/subscribe systems solve a problem inverse to database query processing (i.e., evaluate an event on a set of subscriptions to identify the matching ones).

Information dissemination services are often "add-ons" to auction sites, shopping sites, or information services (e.g., news, sports, traffic), that allow a subscriber to express interest in certain events and consequently be notified upon the occurrence of the event. For instance, a site offering apartments (rental or sale) may allow a user to submit a detailed subscription constraining location, size, price, and nearby attractions of the ideal apartment. Subscriptions along the lines of

S_i: (*close to* downtown Toronto) AND
(*about* 75 square meters in size) AND
(*no more than* $1000) AND
(*close to* major grocery shopping)

are convenient, but to the best of our knowledge, not supported by any dissemination service or publish/subscribe model. The key difference between this kind of subscription and conventional subscriptions is the use of approximate predicates to constraint the subscriber's interest.

Profile-driven location-based service applications that alert mobile users according to events that match the users' profiles and locations are another example that would benefit from processing approximate, rather than crisp information. Todate's location-based enabling technology, for instance, does not allow to continuously and accurately track a users location. A user location, – in many current LBS models is assumed to be provided by the user herself, – gathered through GPS, for instance, is only accurate to a certain degree and it takes a certain time to get the location quote.

All publish/subscribe systems developed to date are based on the assumption that a match between subscriptions stored in the system and an event submitted to the system is either true or false – that is, a subscription matches or does not match. This

Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

partitions the set of subscriptions stored in the P/S system into two disjoint subsets, the matching subscriptions and the not-matching subscriptions. A subscriber is only notified, if her subscription is in the set of matching subscriptions. Often, however, it is very difficult to cast vagueness inherent in real world application scenarios into a crisp model, that establishes exact limits, as we tried to illustrate in the above scenarios.

For these reasons, we think, it is of great advantage to provide a subscription language model and an approximate matching scheme that allows the expression and processing of vagueness. Current P/S system models do not allow this at all. To address this problem we are extending subscription languages to incorporate the expression of vagueness at the language level, to combine this notion with crisp language predicates, and we develop matching algorithms to support the processing of this model in P/S systems. In our model, a subscription will match a given event with a certain degree of confidence. A degree of zero corresponds to the crisp case of not matching at all and the degree of one corresponds to the crisp case of matching, while a degree between one and zero expresses more or less confidence in the established match.

A–TOPSS is the *Approximate Matching-based Toronto Publish/Subscribe System* research prototype that implements this matching scheme and serves us as experimentation platform to evaluate different subscription language models and approximate matching algorithms. In Section 2 we discuss related approaches. In Section 3 we discuss the A–TOPSS research prototype and in Section 4 we describe the software demonstration.

2 Related Work

Much work has been devoted to developing P/S systems and event notification services. These systems are distinguished in the subscription language and the publication model they offer.

The LeSubscribe P/S system focuses on developing pattern-matching algorithms to support subscription languages on an LDAP-like semi-structured data model. In this system [4], a subscription is a conjunction of predicates each of which is a triple (attribute, operator, value). The relational operators supported include $<, \leq, =, \neq, >, \geq$. Elvin [7] is a notification service that targets application integration environments and monitoring of distributed systems. The publication model of ELVIN is similar to the LeSubscribe model, its subscription language also includes boolean expressions over the attributes of an event (e.g., and, or, not, exist(), datatype()) and string operators (=, $\neq$, matches("string"), and regular expression). Siena [3] and Gryphon [2] comprise other examples of P/S research projects that expose a very similar subscription language and publication model. All these systems base the matching semantic on the crisp notion – either a match is established or no match is established, a degree of confidence or degree of match does not exist in these models.

Other approaches apply similarity measures for data retrieval. Fagin [5] associates with each object in a database a degree to which it fullfils a certain characteristic (e.g., a house is highly priced to a certain degree.). Objects are retrieved according to these degrees. Wolski *et al.* [8] use fuzzy membership functions to model event-condition-action rules and their evaluation in active databases. Yan *et al.* [10] suggests an algorithm to compute the similarity between a document and a profile based on the "importance" value of certain attributes. However, all these approaches center around database querying, which is not our intention. A new scheme to do the approximate matching in P/S systems is required.

3 System Description & Architecture

3.1 Subscription language model

Subscriptions model users' interests through boolean combinations of predicates. A predicate is a constraint over a domain of values. For example, the predicate, p_i:(`price` < \$100.00), constraints the attribute `price` to less than 100 dollars (i.e., p_i is true for all permissible values of price that are less than 100 dollars.) We extend this crisp subscription language model to one that can also – or exclusively – express approximate predicates. We experiment with two ways of representing approximation: fuzzy set theory and probability theory.

Both models use similar entities, but express different concepts. Consider a weather condition or the location of a mobile user in a wireless network; we can say that there is a 5% probability of rain tomorrow – or the user is located with a 30% chance in a certain region, – but it is not appropriate to say that it rains to 0.5 degree, unless, of course, we refer to the degree of precipitation.

Fuzzy membership functions are appropriate to model certain kinds of concepts such as color, shape, and taste. For these attributes it is difficult to specify a concrete value because there is no distinct criterion to select a value. Predicates may be expressed more appropriately as "color is red to a degree of 0.8" or "taste is somewhat sweet".

For other kinds of attributes, such as price, age, size, and salary, we have several choices. For one, they can be expressed with predicates of the crisp language model. Approximation is expressed through delimiting the attribute in an interval of

concrete values. For example, "middle age" can be translated to $30 \leq age \leq 55$. This, of course, falls back to the true/false situation, with any warranted differentiation lost. Secondly, these kinds of attributes can be represented by fuzzy membership functions, which determine for a specific value, the degree to which the attribute under scrutiny is defined (e.g., the degree to which a given price is moderate).

We formalize our subscription language as follows: a subscription is a set of predicates, each predicate is a triple containing an attribute, an operator, and a value. There are three kinds of operators: crisp operators (e.g., $>, <, =, \leq, \geq, \neq$) which define crisp predicates; approximate operators (e.g., $\tilde{=}$ modeling approximate equality) which define fuzzy predicates; and probability operators (e.g., $\rightarrow$ modeling a random event) which define probabilistic predicates. For the example mentioned in section 1, the subscription S_i is formalized as follows:

$$\begin{aligned} S_i &= (distance(downtown), \tilde{=}, close) \\ &\wedge (size, \tilde{=}, 75m^2) \\ &\wedge (rent, \leq, \$1000) \\ &\wedge (distance(grocery), \tilde{=}, close) \end{aligned}$$

Note, this is the language as implemented in our research prototype, what is exposed to a user of the system at higher layers may differ (i.e., offer a more natural language-based interface).

For fuzzy predicates a membership function μ, represented as a table or in analytical form, models the predicate's value. The evaluation of the predicate applies the membership function to the corresponding attribute value in the event. More complex fuzzy predicates involve multi-valued membership functions (e.g., fuzzy relations). The degree of match of a subscription is computed from the degrees of match of its predicates according to the common aggregation functions in fuzzy set theory [11]. We use *min* to model *and* and we use *max* to model *or*. Due to this matching semantic many subscriptions match for a given event, but to different degrees. We use a threshold, also referred to as α-cut in fuzzy set theory, to eliminate subscriptions matching only to a small degree. An event e matches a subscription s if and only if for each predicate $p_i : (a_i, =, value)$ of s, there is some attribute-value pair (a_i, v) in e for which the degree of match is greater than α (i.e., $\mu_{value}(v_{a_i}) \geq \alpha$).

3.2 Publication model

In the approximate matching scheme we are proposing, publications represent real life artifacts, such as apartments at a specific location, of a given size and at a fixed price. These artifacts can be adequately represented through the semi-structured models proposed by other P/S systems, we therefore adopt this model. A publication in our system is represented as a collection of attribute-value pairs, denoted as $P_i = \{(A_1, v_1), (A_2, v_2), \cdots, A_n, v_n)\}$, where A_i is an attribute name and v_i is its value.

3.3 System Architecture

A–TOPSS's overall system architecture is similar to the architecture of other P/S systems described in the literature. Its key difference is the subscription language model and the notion of approximate matching it supports. Figure 1 depicts the logical system architecture.

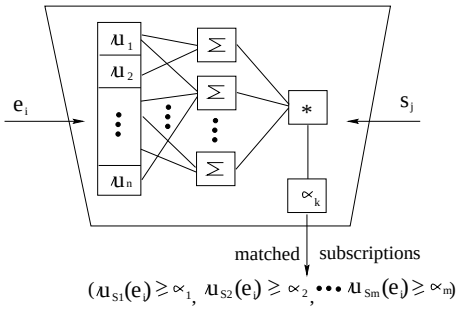

Figure 1: Publish/Subscribe System Architecture

When an event e is submitted, there are three steps to perform matching. First, the attribute-value pairs of e are evaluated on the predicate membership functions μ. (In the figure each function corresponds to an attribute.) Then, subscriptions are evaluated by applying the aggregation operator specified in the subscription over all its predicates. Finally, subscriptions with degrees of confidence greater than a given α-cut value are output as matching subscriptions. Note, these subscriptions can be naturally ordered according to their degree of match.

4 Demonstration

We will demonstrate two scenarios. First, normal system operation, which constitutes an example of the apartment rental market application we have discussed throughout the paper and, second, the experimental system operation to investigate the effects of different representations of approximate predicates in our subscription language model. The demonstration system setup for both scenarios is depicted in Figure 2. All system components shown in this figure will run as different processes on a laptop.

Normal system operation – apartment rental market: This scenario shows the deployment of A–TOPSS as an information dissemination service collocated to an apartment rental market web server. In the demonstration we offer fea-

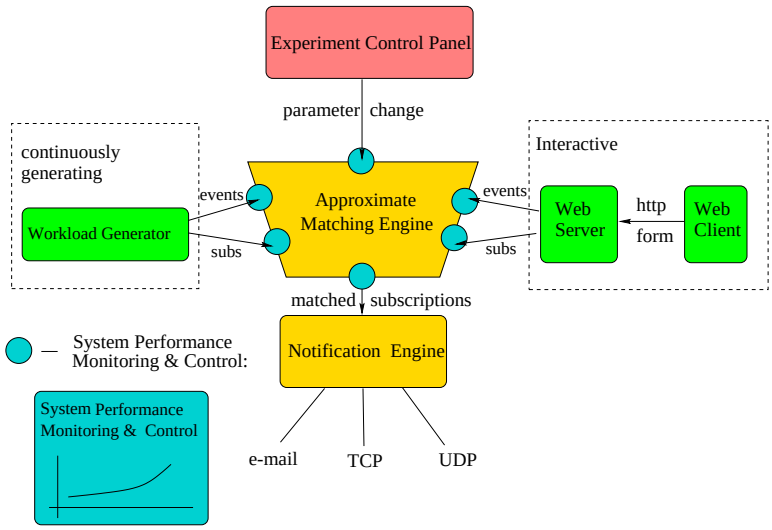

Figure 2: Demonstration Setup

tures for a user to input subscriptions over the web and receive notifications as e-mail messages. In the background we simulate other users and content providers (i.e., landlords or other agencies offering apartments, for instance) through a workload generation process. Our workload generator is script-driven, it can generate new subscriptions and publications at random. This scenario is meant to validate the real world applicability of the approximate matching model we are proposing.

Experimental system operation: Other than in the crisp-model, currently favored by all existing P/S systems, the representation of approximate subscriptions involves many parameter choices. For instance, membership functions can be parameterized or substituted by different functions. The degree of confidence beyond which subscribers should no longer be counted as matches – naturally more subscriptions match, due to the generalized matching semantic, – is another parameter which may influence system operation.

To experiment with such parameters and to demonstrate their effect on system operation, we deploy an experiment control panel (java applet) that can changes these parameters, and a monitoring and control panel (java applet) that observes and displays system metrics (i.e., throughput, number of subscriptions and publications in system queues). These metrics are taken at monitoring and control points indicated in the figure. This scenario aims at experimenting with the approximate matching model to demonstrate and explore its degrees of freedom.

References

[1] Mehmet Altinel and Michael J. Franklin. Efficient filtering of xml documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.

[2] G. Banavar, T.D. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Storm, and D.C. Sturman. An efficient multicast protocal for content-based publish-subscribe systems. In *International Conference on Distributed Computing Systems*, 1999.

[3] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design of a scalable event notification service: Interface and architecture. In *Technical Report CU-CS-863-98*, Department of Computer Science, University of Colorado, August 1998.

[4] F. Fabret, Jacobsen H.A., F. Llirbat, J. Pereira, K.A. Ross, and D. Shasha. Filtering algorithm and implementation for very fast publish/subscribe systems. In *ACM SIGMOD conference*, Santa Barbara, California, USA, May 2001.

[5] R. Fagin. Combining fuzzy information from multiple systems. In *Proc. ACM SIGMOND/SIGACT conf. on Princ. of Database Syst. (PODS)*, Montreal, Canada, 1996.

[6] B. Krishnamurthy R. E. Gruber and E. Panagos. The architecture of the ready event notification service. In *Proceedings of the 19th International Conference on Distributed Computing Systems Middleware Workshop*, Austin, Texas, May 1999. IEEE Computer Society.

[7] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of AUUG97*, Brisbane, Australia, September 1997.

[8] A. Wolski and T. Bouaziz. Fuzzy triggers: Incorporating imprecise reasoning into active database. In *Proceedings of the 14th International Conference on Data Engineering*, 1998.

[9] T. Yan and H. Garcia-Molina. The SIFT information dissemination system. In *ACM TODS*, 2000.

[10] T.W. Yan and H.G. Molina. Index structures for information filtering under the vector space model. In *Proceedings of the International Conference on Data Engineering*, November 1993.

[11] L.A. Zadeh. Fuzzy sets. In *Information and Control*, pages 338–353, 1965.