Uncertainties in Publish/Subscribe System

Ph.D. Research Proposal

Haifeng Liu
Department of Computer Science
University of Toronto

Abstract

In this proposal, we introduce a scalable distributed publish/subscribe system for selective information dissemination. We propose two models A-ToPSS and G-ToPSS for two different matching problems: approximate matching and semantic matching. A-ToPSS aims at processing uncertainty information, while G-ToPSS aims at filtering graph-based metadata. We describe problems left in these two models and make a plan for the future work.

1 Introduction

With the proliferation of pervasive computing devices, the integration of network access technologies (mobile wireless and Internet), it is becoming increasingly easier for non-computer oriented users to publish information on the Internet because of myriads of user-friendly tools that now exist. As a result, the information on the Internet increases dramatically. Today most information dissemination services, e.g., RSS aggregator, use a pull-based architecture where users have to actively pull from a web site to find out what they need. As the amount of information on the Internet is continuously increasing, the pull-based architecture is not going to scale because it not only consumes unnecessary resources, but also becomes difficult to ensure timely delivery of updates.

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, which anonymously interconnects information providers with information consumers in a distributed environment. Information providers publish information in the form of publications and information consumers subscribe their interests in the form of subscriptions. The publish/subscribe system performs the matching task and ensures the timely delivery of publications to all interested subscribers.

Publish/subscribe has been well studied and many systems have been developed supporting this paradigm. Existing research prototypes, include, among others, Gryphon [7], LeSubscribe [13], PADRES [14] and ToPSS [19, 27, 30, 9, 17]; industrial strength systems include various implementations of JMS [16, 24], the CORBA Notification Service [25], and TIBCO's Tib/Rendezvoud product [31] and Web Services Notification [6]. There are some open problems for current publish/subscribe systems.

First, common to all the current systems is the crisp matching semantics, which means that neither subscribers nor publishers can express uncertain or partial information. However, in many situations exact knowledge to either specify subscriptions or publications is not available. In these cases, the uncertainty about the state of the world has to be cast into the crisp data model that defines absolute limits. Moreover, for a user of the publish/subscribe system it may be much simpler to describe the state of the world with vague and uncertain concepts. That means in an approximate manner.

In a selective information dissemination context, for instance, users may want to submit subscriptions about a book whose constraints on price is “cheap”. On the other hand, information providers may not have exact information for all items published. In a second-hand market, a seller may not know the exact age of a antique vase so that she can just describe it as an “old” vase. Temperature and humidity information collected by sensors are often not fully precise, but only correct within a certain error interval around the value measured. It would be better to publish imprecise information, rather than a wrong exact value, if such publish/subscribe capabilities were possible.

Also, uncertainty exists for location-based information dissemination services. Owing to the movement of the mobile user and transmission delay, approximate location would be more appropriate rather than assessing the precise location of each user. To date location-based enabling technology, for instance, does not allow to continuously and accurately track a user’s location. The user location is in many current LBS models assumed to be provided by the user herself. If it is gathered through GPS it is only accurate to a certain degree and can only be gathered so and so often (e.g., it takes around 30 seconds to get the first location quote.)

All publish/subscribe systems developed to date are based on the assumption that a match between a subscription 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 partitions the set of subscriptions stored in the publish/subscribe 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. However, it is very difficult to cast the vagueness inherent in real world 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 publish/subscribe data model and an approximate matching scheme that allows the expression and processing of uncertainty for both subscriptions and publications.

Secondly, most current publish/subscribe systems are based on content matching mechanism and do not consider the structure or semantics of the data. In systems such as Elvin [29], LeSubscribe [13], PADRES [14] and ToPSS [19, 27, 30, 9, 17] and CREAM [11], the publication and subscription model is based on attribute value pairs. There is no structure and semantic information associated at all. Systems as described in [15] can process XML publications and XPath subscriptions. However, these approaches are based on a tree index structure to support filtering of XML documents over XPath quries. These approaches do not support the filtering of graph-structured data, which is more general than tree-structured data. Graph-based data is used in many emerging semantic web applications(e.g., blogs, wiki updates).

For example, RDF (Resource Description Framework) is used to represent information and to exchange knowledge on the web [2]. A RDF document is represented as a directed labelled graph. An application for filtering RDF documents is RSS (RDF Site Summary) [5], a RDF-based language for expressing content changes on the web, which has grown considerably in popularity. RSS is so versatile that any kind of content changes can be described (e.g., web site modifications, wiki updates, history of a source code from a versioning software (e.g., CVS)).

However, existing RSS systems do not scale well. In current RSS delivery systems, multiple RSS aggregators continuously *poll* numerous RSS feed sites [1], which results in that websites hosting popular RSS feeds can be significantly overloaded with useless network traffic. The publish/subscribe system offers a solution to avoid the inherent polling-based design and provides scalability, as more users publish and expect to stay current with changes submitted by others.

Use of RDF also makes it possible to use ontologies built on top of RDF using languages such as RDFS [3] and OWL [4] to process syntactically different, but semantically-equivalent information.

Existing work as in Racer [15] is a publish/subscribe system based on a description logics inference engine and can be used for RDF/OWL filtering. But racer does not scale well, the matching time for Racer is in the order of 10s of seconds even for very simple subscriptions (e.g. $retrieve(book)$).

Considering both expressiveness and scalability, it would be better if a publish/subscribe system model is developed to support large-volume graph-based content distribution from diverse sources and allows the use of ontologies to specify class taxonomy as semantic information about the data.

Up to now, we discuss two matching problems in publish/subscribe system: approximate matching and semantic matching. These two problems are discussed with respect to a centralized architecture. To provide better scalability, several previous work have proposed a distributed architecture that uses a network of brokers [10, 8, 26]. In a distributed publish/subscribe, there are three main entities: publishers, subscribers and brokers. Publishers send all data to a broker (or a network of brokers). Subscribers register their interest in receiving certain data with a broker. The broker records (persists) all subscriptions in the system, matches them against incoming publications and notifies the corresponding subscribers. Other than matching problem, information routing is another important problem in a distributed publish/subscribe system. Routing directs forwarding, the passing of pieces of information from their source toward their ultimate destination through intermediary nodes (called routers). In a content-based distributed system, messages from publishers (publications) do not contain any address; instead, they are routed through the system based on their content.

A network of brokers can be formed to create a content-based routing system. To avoid propagating subscriptions to all brokers in the network and to improve the matching efficiency, *Covering* and *merging* are two optimizations for content-based routing [18], which "combine" related subscriptions together and propagate the combined subscription to reduce the network traffic and increase the efficiency. Therefore, how to define correspondent covering and merging for the above two models aiming at approximate matching and semantic matching, respectively. Moreover, new routing algorithms are needed for the routing computations. Especially considering the popularity of semantic web, how to define routing computations involving ontologies in a distributed system gives rise to another interesting problem.

The Middleware System Research Group at the University of Toronto is working on the Toronto Publish/Subscribe System (ToPSS) family of research projects for distributed event-based processing. A-ToPSS and G-ToPSS are two projects among them. My previous work has described a publish/subscribe system (A-ToPSS) supporting uncertainties including the language model, the matching semantics, the data structure and the algorithms underlying the computations for A-ToPSS in a distributed content-based routing [21, 19, 20]. Similar problems for Graph-based Publish/Subscribe System (G-ToPSS) have also been developed [28]. However, these two models haven't not been investigated completely and can be improved. In this proposal, we will discuss the directions for improvement, explore their feasibility and make a plan for the next step.

2 Research Plan

The above two models can be improved from the following aspects.

1. In the A-ToPSS model, we use fuzzy sets to represent an uncertain constraint over all possible values the attribute can take. A fuzzy set is represented by a membership function. Figure 1 shows an example of a possible membership function for the fuzzy set of "warm". To choose an appropriate membership function for a user to define his own specification is a challenge. It would be better to provide default membership function representations for uncertain notions

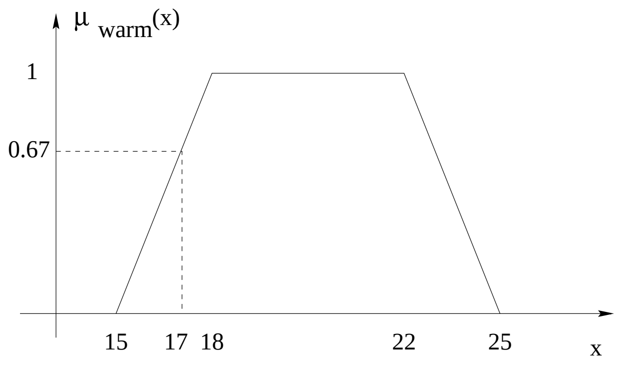

Figure 1: Possible Membership function of "warm" for temperature in $^\circ C$

where the default functions represent the perspective of the majority of users for a given attribute.

2. In the distributed G-ToPSS model, currently we only consider the covering, merging and intersecting algorithms based on the structures of subscription graph patterns. However, the complete G-ToPSS model involves the support of ontologies. Therefore, we need to extend the routing computations for ontologies. New covering, merging and intersecting algorithms for ontologies should be developed for distributed routing of graph-based metadata.

3. Many methods have been proposed to process imperfect information. Besides fuzzy set and possibility theory, probabilities are extensively used to model uncertainties. What is the difference between these two methods and how to compare their performance is the third problem we will consider.

In this section, we will discuss the first two problems in detail and the third problem is under investigation.

2.1 Membership Function Mining

The A-ToPSS model offers its users great flexibility and leaves room for tuning a wide range of default parameters. It is often a challenge to select the right membership function parameterization, the exact number of membership functions to represent one dimension, the appropriate aggregation functions or the right thresholds. Users may like to define their own specifications according to the information of other people's requirements. It will be more convenient if we could provide them with the aggregated knowledge of current data in the system. For example, a user wants to get to know the average price for a second-hand car and buy one with a price relative to the notion of cheap in the system. Therefore, his specification for the price may vary according to the average price. If the average price is 10K dollars, he will define his expectation between 8K to 11K. However, if the average price is 5K, his expected price will decrease to $[4k, 6k]$.

A-ToPSS is used in a context where many subscribers (potentially millions) seek the right information. Consequently, much information about what defines certain concepts in specific domains is readily available, such as an "average understanding" of what constitutes a "cheap" price of a popular electronics gadget available in an online auction. If this information could be exploited, better default parameter choices could be determined for subscribers and publishers of such a system.

In A-ToPSS we experiment with a clustering-based approach that determines default value settings from historical data (i.e., from past subscriptions and publications). We demonstrate our approach on real data traces that we have collected from an online auction site.

In the clustering analysis we do not differentiate between subscriptions or publications, but mine for default parameterizations of the membership functions underlying both entities. This is possible due to the link of a fuzzy set and a possibility distribution, explained in further detail in [21]. The mined parameterization is then used to provide default values for imperfect concepts. Currently, we focus on estimating the number of concepts that define one dimension and their function representation. For example, the dimension price, could be represented by the concepts ”cheap”, ”fair” and ”expensive” or by four concepts, adding, for instance, ”luxurious”. Each of these concepts is represented by a parameterized membership function, which can be adapted to a specific understanding by modifying its parameters.

There are two difficulties in membership function mining. One is that the feasibility of learning depends on the distribution of the data in data set. To learn parameters for membership functions is a complicated task. There are two steps. First, the number of membership functions to represent the characteristics of one attribute should be decided. Next, the parameters to represent the shape of the function is learned for each membership function. The first problem is a clustering problem and the second is a regression problem. Both problems depend on the distribution of the data. The solutions only exist when the data set can be classified into several categories and there is a pattern for each category. Therefore, the first task is to verify that the data can be classified and data patterns exist in the real world. We will find a real world data set which is applicable to our model and can be used for membership function mining.

Given the data set, many learning methods have been proposed for solving the clustering problem and the regression problem. For clustering, we can apply K-means algorithm [22] or the DBscan algorithm [12]. K-means is a partitioning clustering algorithm. Using K-means, the number of clusters needs to be defined beforehand and assign each piece of data into a cluster. The number of clusters is not required to be defined beforehand in the DBscan algorithm. DBscan it is a density-based algorithm for discovering clusters in a large data set. Clusters are generated according to the density in one cluster. However, the density and the minimum number of neighbors are required to be given at the beginning. The regression problem is to find a function f such that

$$\sum_{i=1}^{m} |f(x_i) - y_i|^2$$

is minimized. Neural network learning method is a good fit for this problem [23].

Finally, the evaluation of the membership functions learned from historical data is addressed. This is to evaluate whether the functions we get really reflect the common understanding of most people for a given notion; and whether the default parameters provide the users a better understanding of the information currently in the system and help the users to define their interests accordingly. In order to answer these questions, evaluation has to be performed in an interactive process where humans are involved to assess the correctness of the membership functions.

2.2 Semantic Matching and Routing Computation

There are two levels of semantic matching and routing in publish/subscribe systems. One is for single broker or a set of brokers where their subscription and publication language models are the same, but the terms used in the language are different. For example, two subscriptions about vehicles are subscribed from two brokers separately. Both subscriptions are in a predicate-based form where each predicate is a triple consisting of the attribute, operator and value. One uses “car” as the attribute name and the other uses "automobile". The semantics among different terms need to be exploited for semantic matching and routing. The other level is in the semantic mapping between different language models. Similar to the data integration in database, in publish/subscribe systems, there also exists a problem to integrate the subscription and publication information among different brokers. For example, one broker use a content-based publish/subscribe model where the subscriptions and publications are expressed by a set of predicates. Another broker accepts subscriptions and publications in the form of XML document and XPath. To merge these subscriptions together, we need to exploit the semantic mapping between two language models. In this paper, we only consider the semantic matching and routing in the first level.

The components in a publish/subscribe system, especially a distributed environment, are a priori decoupled, anonymous, and do not necessarily "speak" the same language. However, different language may refer to the same entity or have the same meaning in the real world. This problem can be solved by a semantic publish/subscribe system, which based the matching on semantics, not syntax of the language. The main functionality that a semantic publish/subscribe system needs to provide is best illustrated using an example. For example, suppose we have a job-finder application used by interested companies to look for potential employees. If there is a company recruiter interested in candidates who graduated from a certain university, with a PhD degree and with at least 4 years of professional experience, then the recruiter would subscribe to the following:

$$S : (university = Toronto) \wedge (degree = PhD) \wedge (professional\_experience \geq 4).$$

A prospective candidate would enter the following event using the job-finder application:

$$E : (school, Toronto)(degree, PhD)(work\_experience, true)(graduation\_year, 2000).$$

Then the pub/sub system running the job-finder application should match the event and the subscription above, and send the resume of the candidate to the company recruiter since $(university = Toronto)$ and $(school, Toronto)$ refer to the same entity and $(graduation\_year, 1990)$ implies the candidate have 6 years of professional experience.

Similarly, in a distributed semantic publish/subscribe system, covering and merging algorithms can be applied to the subscriptions with the related semantics though they have no relation from the aspect of the language syntax. From example, we have two subscriptions that both are interested in buying a used car. These two subscriptions are subscribed from different brokers and use different languages in the following forms:

$$S_1 : (car = honda) \wedge (age \leq 5)$$

$$S_2 : (automobiles = ACCORD) \wedge (bought\_year \geq 2003)$$

From the aspect of syntax, there is no relation between S_1 and S_2. However, S_1 covers S_2 if the semantics of the terms used in the subscriptions are considered.

As explained in [27], usually, there are three semantics relationships: synonymous, concept hierarchy (is-a) and computation mappings. As in the above example, "car" and "automobile" are synonyms and "$bought\_year \geq 2003$" is equivalent to "$age \leq 3$" by additional computations.

In a semantic publish/subscribe systems, subscriptions and publications can be first processed by transforming to uniform semantic equivalent forms, then matching, covering or merging algorithms are applied to uniform equivalents.

The G-ToPSS model supports a special semantic matching using ontology for graph-based metadata. A RDFS class taxonomy with *is-a* relationship is the semantic information about a $subject$ or an $object$ that is available in the G-ToPSS ontology. Multiple inheritance is allowed and the only restriction on the taxonomy is that it must be acyclic. All instances of a class are listed in the taxonomy. Alternatively, this information can be specified in the RDF graph using a *type* property, but for simplicity we have opted to include this information in the taxonomy.

When filtering, a subscription is matched if and only if both the content and the ontology constraints of each node are satisfied. The ontology constraint for each node is expressed by ($?x$ *op* v) where $?x$ refers to a variable node and v is a class. *op* is *is-a* operators and is in the form of $=$, $\leq$ and $\geq$ but with alternative semantics. $\leq$ means the instance value of $?x$ is the descendant of the class v. $\geq$ means that the instance value of $?x$ is the ancestor of class v. $=$ means that the value of $?x$ is the direct instance of class v (i.e., a child of v). For the class taxonomy filtering constraint ($?x$ *op* v), we need to check the descendent-ancestor relationship between the specific instance value $?x$ and the class v by traversing the taxonomy tree. For example, in a subscription to search for a certain paper, the paper node has a taxonomy constraint that it has to belong to publications. This node can be matched by node "Arno's paper $\sharp 17$" since it is descendant of class "Publication."

In a distributed G-ToPSS model, the subscription cover relation should be defined to include ontology cover relation to compromise semantics covering and merging algorithms. Ontology cover relation can be defined according to the ancestor-descendent relationship between two classes and the similarity measurement for merging can be defined as the distance of two classes in the ontology tree.

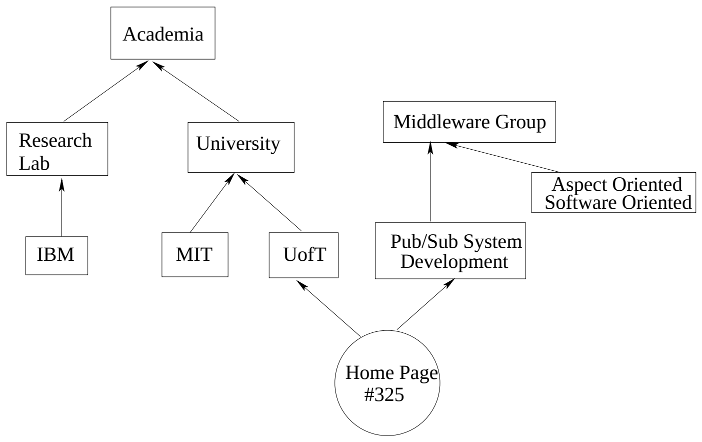

Figure 2: Example taxonomy

In Figure 2, we show an example of a class taxonomy about an academic webpages system. Class "Academia" includes twosubclasses: "Research Lab" and "University". Class "Middleware" includes "Pub/Sub" and "AOP" two subclasses. The document instance "Home page $\sharp 325$" belongs to both "UofT" and "Pub/Sub".

According to the above ontology, class taxonomy constraint ($?x$, $\leq$, $university$) will cover the constraint ($?x$, $\leq$, $UofT$) since $university$ class is the ancestor of $UofT$ class. And constraints ($?x$, $=$, $UofT$) and ($?x$, $=$, MIT) can be merged into one constraint ($?x$, $\leq$, $university$) where use the ancestor of both $UofT$ and MIT as the class constraint.

Together with existing covering and merging algorithms for only structure information of G-ToPSS model, a completed routing computations will be proposed to support distributed filtering of graph-based structured data with ontology support.

3 Status

The following results have been achieved over the past three years.

1. An approximate publish/subscribe model to express uncertainties in both subscriptions and publications has been proposed. This model applies when exact information to specify publications and subscriptions is not available. Fuzzy set theory and possibility theory are used to represent uncertainty in this model. An approximate matching relation between publications and subscriptions is defined. Two matching algorithm exhibiting different trade-offs are designed to solve the approximate publish/subscribe matching problem.

2. A *covering* and *merging* scheme is developed to extend the approximate publish/subscribe model to a content-based routing model (i.e., a distributed publish/subscribe model.) Efficient algorithms are proposed and implemented supporting approximate content-based routing.

3. To support semantic matching, a graph-based publish/subscribe architecture for dissemination of RDF data, such as RSS, is proposed. In this model publications and subscriptions are graph-structured objects. A matching relation is defined on these graph structures. As an application, the fast filtering of RDF metadata, such as publications extracted from RSS feeds is demonstrated.

4. A *covering* and *merging* scheme is developed to extend the graph-structured publish/subscribe model to a content-based routing model (i.e., a distributed publish/subscribe model.) Algorithms for computing *covering* and *merging* relations over graph-structured data are proposed and evaluated. The approach is applied to the filtering of RDF-based metadata, such as RSS feeds, in distributed publish/subscribe architectures.

Based on this proposal, the following tasks will be carried out to complete this thesis.

1. Develop an algorithm to mine default parameters for membership functions based on statistical information and historical data.

2. Design an experiment to evaluate the practicality of the membership function mining algorithm and assess the advantage of the approximate publish/subscribe system over traditional publish/subscribe models. An effort will be made to use real-world data for the comparison. The source of this data is uncertain at this point.

3. Exploit the semantic covering and merging operations on ontologies to complete the work of semantic matching and routing.

4. Investigate the applicability of probabilities in the publish/subscribe model. This will result in a probabilistic publish/subscribe model. This model represents an alternative for the representation of uncertainties. A comparison to the traditional, the approximate, and the graph-structured model will be made.

References

[1] *http://www.rss-specifications.com/aggregator-how-to.htm*.

[2] *http://www.w3.org/RDF/*.

[3] *http://www.w3.org/TR/2000/CR-rdf-schema-20000327/*.

[4] *http://www.w3.org/TR/owl-features/*.

[5] *web.resource.org/rss/1.0/spec*.

[6] *Web Services Notification (WS-Notification), Version 1.0.* 2004.

[7] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *19th IEEE International Conference on Distributed Computing Systems (ICDCS)'99*, 1999.

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

[9] I. Burcea and H.-A. Jacobson. L-ToPSS - push-oriented location based services. In *Proceedings of the 2003 Workshop on Technologies for E-Services*, Lecture Notes in Computer Science. Springer, 2003.

[10] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, Aug. 2001.

[11] M. Cilia, C. Bornhoevd, and A. P. Buchmann. CREAM: An Infrastructure for Distributed Heterogeneous Event-based Applications. In *Proceedings of the International Conference on Cooperative Information Systems*, pages 482–502, 2003.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, August 1996.

[13] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *The 20th ACM SIGMOD International Conference on Management of Data*, 2001.

[14] E. Fidler, H.-A. Jacobsen, G. Li, and S. Mankovski. The padres distributed publish/subscribe system. In *International Conference on Feature Interactions in Telecommunications and Software Systems(ICFI'05)*, Leisester, UK, 2005.

[15] V. Haarslev and R. Moller. Incremental Query Answering for Implementing Document Retrieval Services. In *Proceedings of the International Workshop on Description Logics*, 2003.

[16] M. Happner, R. Burridge, and R. Sharma. Java message service. October 1998.

[17] H. K. Y. Leung. Subject space: A state-persistent model for publish/subscribe systems. In *Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, page 7. IBM Press, 2002.

[18] G. Li, S. Hou, and H. A. Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. *International Conference on Distributed Computing Systems (ICDCS'05)*.

[19] H. Liu and H.-A. Jacobsen. A-ToPSS - a publish/subscribe system supporting approximate matching. In *Proceeding of the 28th International Conference on Very Large Data Bases*, Hong Kong, August 2002.

[20] H. Liu and H.-A. Jacobsen. A-ToPSS - a publish/subscribe system supporting imperfect information processing. In *the 30th International Conference on Very Large Data Bases*, Toronto, Canada, 2004.

[21] H. Liu and H.-A. Jacobsen. Modeling uncertainties in publish/subscribe system. In *Proceedings of the 20th International Conference on Data Engineering*, Boston, USA, April 2004.

[22] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[23] T. Mitchell and M. Hill. *Machine Learning*. McGraw-Hill College, 1997.

[24] R. Monson-Haefel and D. A. Chappell. Java message service. In *O'Reilly*, 2001.

[25] Object Management Group. Notification service specification, version 1.0.1. August 2002. http://www.research.att.com/ ready.

[26] O. Papaemmanouil and U. Cetintemel. Semcast: Semantic multicast for content-based data dissemination. In *Proceedings of International Conference on Data Engineering*, 2005.

[27] M. Petrovic, I. Burcea, and H.-A. Jacobsen. S-ToPSS - a semantic publish/subscribe system. In *Proceedings of the 29th International Conference on Very Large Data Bases*, Berlin, Germany, September 2003.

[28] M. Petrovic, H. Liu, and H.-A. Jacobsen. G-ToPSS - fast filtering of graph-based metadata. In *the 14th International World Wide Web Conference*, Chiba, Japan, May 2005.

[29] P. Sutton, R. Arkins, and B. Segall. Supporting disconnectedness - transparent information delivery for mobile and invisible computing. In *2001 IEEE International Symposium on Cluster Computing and the Grid*, 2001.

[30] D. Tam, R. Azimi, and H.-A. Jacobsen. Building content-based publish/subscribe systems with distributed hash tables. In *International Workshop On Databases, Information Systems and Peer-to-Peer Computing*, Berlin, Germany, September 2003.

[31] TIBCO Inc. Tibco/rendezvous concepts. October 2000.