Extending Siena to support more expressive and flexible subscriptions

John Keeney, Dominik Roblek, Dominic Jones, David Lewis, Declan O’Sullivan
Knowledge & Data Engineering Group (KDEG)

University of Dublin, Trinity College, Ireland

{ John.Keeney | Roblekd | Jonesdh | Dave.Lewis | Declan.OSullivan }@cs.tcd.ie

ABSTRACT

This paper defines and discusses the implementation of two novel extensions to the Siena Content-based Network (CBN) to extend it to become a Knowledge-based Network (KBN) thereby increasing the expressiveness and flexibility of its publications and subscription. One extension provides ontological concepts as an additional message attribute type, onto which subsumption relationships, equivalence, type queries and arbitrary ontological subscription filters can be applied. The second extension provides for a bag type to be used that allows bag equivalence, sub-bag and super-bag relationships to be used in subscription filters, possibly composed with any of the Siena subscription operators or the ontological operators previously mentioned. The performance of this KBN implementation has also been explored. However, to maintain scalability and performance it is important that these extensions do not break Siena’s subscription aggregation algorithm. We also introduce the necessary covering relationships for the new types and operators and examine the subscription matching overhead resulting from these new types and operators.

Categories and Subject Descriptors

C.2.2 Network Protocols: Routing protocols, I.2.4 Knowledge Representation Formalisms and Methods: Semantic Networks

General Terms

Performance, Experimentation.

Keywords

Publish-subscribe, Content-based Networks Knowledge-based Networks, Ontologies, Semantics, Bags.

1. INTRODUCTION

Publish-subscribe event systems route event messages from producers to consumers that have expressed an interest in a message type in a subscription. While some systems required publisher-consumer agreement on message types, Content-based Networks facilitates much looser coupling through subscriptions that express interest only in some attributes of the event message [1, 2, 3]. Content-based Networks (CBN) have formed around the necessity to match a varying subscriber base to that of publication creators. This achieves a “de-coupling” of the parties involved in the communication process and allows for message routing to be conducted based on who is interested in a particular message, through a routing table compiled from subscription filters. The underlying routing structure over which messages pass allow for a message inserted on one side of the network to propagate across the network based on matching filters (subscriptions) until each and every client interested in the message has been delivered the message.

The filters which match these subscriptions to publications are constructed using a set of operators, the range of these operators determines the type of pub/sub network in which the message is being sent. Content-based Filtering, which is the core message matching algorithm within Content-based Networks is described by Mühl et al in [4] as “Filters which are evaluated against the whole contents of notifications” in this case notifications can be thought of as publications. Notifications are only forwarded to a node when the contents of the filter, which is made up from a set of constraints, matches against the message’s contents. This allows for a more flexible message format and increases the de-coupling between clients within the network. De-coupling in this sense is increased through the ability for sections of the message to match as opposed to complete message matches, and so the publishers and subscribers do not need to have previously agreed on a common format or structure.

However, open standards for CBNs have been slow to emerge due to the difficulty in reaching a general compromise between the expressiveness of event attribute types and subscription filters and the need to both match these efficiently at CBN nodes and to efficiently maintain routing tables. The latter must maximize the multicast efficiencies made possible by aggregating/covering subscriptions with ones that match the same or wider range of messages. In the Siena CBN [1] subscription covering is achieved by restricting attribute types and subscription filters to simple number, string and boolean types and a set of transitive operators to filter them (i.e. greater/less than, super/sub string etc.).

The particular flavour of CBN which is investigated in this paper is an extension of the Java Siena CBN middleware [5]. A Siena notification is a set of typed attributes. Each attribute is comprised of a name, a type and a value. The current version of Siena supports the following types: String, Long, Integer, Double and Boolean. A Siena subscription is a conjunction of filtering constraints, constraints being comprised of the attribute name, an operator, and a value. A subscription matches a notification, if the notification matches to all filtering constraints of the subscription’s filter. The notification is then delivered to all of the clients that submitted those subscriptions that match against that notification. Siena also discovers coverings between filters to aggregate subscriptions and so optimise its routing table. A filter covers another filter, if all notifications selected by the latter are also selected by the former. [1] defined three basic types of Siena topology: hierarchical client/server, acyclic peer-to-peer, and general peer-to-peer. All topologies provide the same functionality, however they differ in non-functional features, like time complexity, scalability and fault tolerance. The specific extension to Siena examined introduces two extensions to the existing type set of the hierarchical Java version [5], supporting bags and ontological types.

2. THE BAG EXTENSION

According to [6], a bag (also called multiset) is a set-like object in which order is ignored, but multiplicity is explicitly significant. Therefore, bags $\{1, 2, 3\}$ and $\{2, 1, 3\}$ are equivalent, but $\{1, 1, 2, 3\}$ and $\{1, 2, 3\}$ differ. A bag differs from a set in that each member has a multiplicity indicating how many times it is a member.

As presented in [7] a bag value can contain any valid Siena values, including other bag values. A bag is not allowed to contain itself, either directly or indirectly via other bags. Elements of a bag do not need to be of a uniform type. In the extension presented here bags are first order members of the Siena type set. They can appear in notifications, as well as in subscription filters, like any other Siena type. Siena advertisements, which are part of the theoretical Siena model, are not supported in the hierarchical version of Siena, but should they be, bag type should work seamlessly with them.

Therefore, some examples bags are: $\{3, 345, 27, 35, 3476, 0, 27, 27\}$, $\{\text{"Ljubljana"}, 2, \text{"Ljubljana"}, 3.14159\}$, $\{\text{"Ljubljana"}, \text{"Vienna"}, \text{"Amsterdam"}, \text{"Dublin"}\}$. Since a set is a bag where all elements have a cardinality of one, this extension also implicitly supports sets. The bag extension adds simple binary bag operators and composite binary bags operators.

2.1 Simple Bag Subscription Operator

The simple operator supports the three well-known binary bag relations: equal, subbag, and superbag.

Two bags A and B are equal, $A=B$, if the number of occurrences of each element in A or B is the same in each bag. For example the bags $\{b, o, o, k\}$ and $\{b, o, k, o\}$ are equal bags but $\{b, o, o, k\}$ and $\{b, o, k\}$ are not equal bags.

For the subbag relationship, we can define A to be a subbag of B, $A \subseteq B$, if the number of occurrences of each element χ in A is less than or equal to the number of occurrences of χ in B. For example $\{b, o, k\} \subseteq \{b, o, o, k\}$ but $\{b, o, r\}$ is not a subbag of $\{b, o, o, k\}$. It follows from the definition of subbag that bags A and B are equal bags, if and only if $A \subseteq B$ and $B \subseteq A$. Another way to describe the subbag relationship is to say that if $A \subseteq B$ then bag B includes or contains A.

The superbag relationship is the inverse of the sub bag relationship. If B is a subbag of A, $B \subseteq A$, then A is a superbag of B, $A \supseteq B$.

Note, a bag can also contain other bags. For example $\{\{g, o, o, d\}, \{b, o, o, k\}\} \supseteq \{\{b, k, o, o\}\}$. All three simple bag relations, namely equal, subbag, and superbag, are transitive and reflexive. The transitivity and reflexivity of simple bag relations follows from the transitivity and reflexivity of numerical *equal* and *less than or equal* relations that were used in the definition of the simple bag relations.

The main advantage of the bag type and bag operators lies in the ability to define much more expressive and flexible subscriptions. For example without the use of bags, multiple filter constraints in a single subscription filter are joined by conjunction (using the boolean AND operator), a disjunction of constraints (using the boolean OR operator) could only be specified using multiple distinct subscriptions. Using bags and the subbag operator the subscriber can create a filter where matching values should be

drawn from a bag of possible values for example. A particularly useful example of this is in Keyword matching.

2.1.1 The Covering Relationships between Simple Bag Operators

Siena optimizes its routing tables by aggregating event filters. The aggregation rules are derived from coverings between individual event filters. If the filter A always matches all or more events that are matched by the filter B, then A covers B. In order to preserve correctness and efficiency of Siena, filter covering relationships for the new bag operators must be properly implemented.

Consider two filtering constraints C_1 and C_2 , such that C_1 is given as $x \Phi P$, and C_2 is given as $x \Phi Q$, where x is a named attribute of type bag, P and Q are the bag values, and the Φ operator is one of: subbag, superbag or equal bag. Covering relationships between simple bag operators are straightforward and presented in the table 1. (Here "equals" refers to the "equal bag" operator).

A	B	A covers B exactly when
x equals P	x equals Q	Q equals P
x superbag P	x equals Q	Q superbag P
x subbag P	x equals Q	Q subbag P
x equals P	x superbag Q	Never
x superbag P	x superbag Q	Q superbag P
x subbag P	x superbag Q	Never
x equals P	x subbag Q	Never
x superbag P	x subbag Q	Never
x subbag P	x subbag Q	Q subbag P

Table 1: Covering relationships between simple bag operators

2.2 Composite Bag Subscription Operator

The composite bag relation is also a binary relation over bags, but is composed of (i) a simple binary bag relation over the bags and (ii) a sub-relation over the bags' elements.

Suppose Φ is a simple binary relation over bags (Φ is a simple bag operator), and λ is an arbitrary binary relation (λ is any non-bag subscription operator). Bag P is Φ -related to bag Q when sub-relation λ is applied between the elements of P and Q, written as $P \Phi_\lambda Q$, if and only if there exist some sequences X and Y, (X is some ordered list of elements from P, and Y is some ordered list of the elements from Q), so that all of the following statements are true:

1. P is Φ -related to $\tau(X)$, where $\tau(X)$ denotes the bag of all elements in sequence X.
bag P is Φ -related to X when X is expressed as a bag
2. $\tau(Y)$ is Φ -related to Q, where $\tau(Y)$ denotes the bag of all elements in sequence Y.
when Y is expressed as a bag, that bag is Φ -related to Q
3. $|X| = |Y|$
sequences X and Y have the same number of elements
4. $\forall i \in \mathbb{N}$ (natural number), $i < |X|$, X_i is λ -related to Y_i .
for (int i = 0; i < |X|; i++)
X.elementAt(i) is λ -related to Y.elementAt(i)

We call relation Φ the primary relation of the composite bag relation, and relation λ the sub-relation of the composite bag relation. So for bags P and Q , simple bag relation Φ , and any relation λ , $P \Phi_\lambda Q$ means that P is Φ -related to bag Q when sub-relation λ is applied. If the composite Φ_λ relation is being used as a subscription operator we call Φ the primary bag operator of the λ the suboperator.

The bag of integers $\{1, 1, 2, 3, 4\}$ is a superbag of $\{2, 4, 3\}$ using the default “equals” ($=$) sub-relation, i.e. $\{1, 1, 2, 3, 4\} \supseteq \{2, 4, 3\}$ (for every element in the second bag, there exists an element in the first bag that is equal to the element, with no reused elements in either bag).

The bag of integers $\{1, 2, 3\}$ is an equal-bag of $\{2, 3, 4\}$ using the “less than” ($<$) sub-relation (for every element in the second bag, there exists an element in the first bag that is less than the element, with no unused or reused elements in either bag), so $\{1, 2, 3\} =_< \{2, 3, 4\}$.

The bag of Strings $\{\text{“ood”}, \text{“boo”}\}$ is a sub-bag of $\{\text{“a”}, \text{“good”}, \text{“book”}\}$ using the “substring” (substr) sub-relation (for every element in the first bag, there exists an element in the second bag such that the element in the first bag is a substring of the element in the second bag, with no reused elements in either bag), so $\{\text{“ood”}, \text{“boo”}\} \subseteq_{\text{substr}} \{\text{“a”}, \text{“good”}, \text{“book”}\}$.

Note, the simple bag operators defined in the previous sections can be defined as composite bag operators using the default “equals” ($=$) sub-relation. More generally, any bag operator Φ is equivalent to the $\Phi_=_$ composite bag relationship.

Let Φ be a simple binary relation over bags, and λ a binary relation over bag elements. (i.e. Φ_λ , Φ is the primary bag operator, λ is the sub operator). If λ is transitive, then Φ_λ is also transitive. If λ is reflexive, then Φ_λ is also reflexive.

Another interesting observation is that if $A \subseteq B$ then $B \supseteq A$. More generally, if Φ is a simple binary relation over bags, λ is a binary relation over bag elements, Φ^{-1} is the inverse relation of Φ , and λ^{-1} is the inverse relation of λ , then $P \Phi_\lambda Q$ exactly when $Q \Phi^{-1}_{\lambda^{-1}} P$.

A composite bag operator may have another composite bag operator as its sub operator. For example given

$$V = \{\{\}, \{0, 0\}, \{1, 2, 3, 4\}\}$$

$$W = \{\{8\}, \{0\}, \{1, 1, 1\}, \{2, 3, 4, 5, 6\}\}$$

then

$$V \subseteq_{(\subseteq)} W$$

since

$$\{\} \subseteq \{8\}, \{0, 0\} \subseteq \{1, 1, 1\},$$

$$\{1, 2, 3, 4\} \subseteq \{2, 3, 4, 5, 6\}, \text{ and bag } \{0\} \text{ in } W \text{ is unused.}$$

Compared to simple bag relations, composite bag relations make looser comparisons of bags possible. They allow for “inexact” matches that would not be possible should we use only simple bag relations.

2.2.1 The Covering Relationships between Composite Bag Operators

Composite bag operators have much more complex semantics and for this reason the covering relationships between them are harder to discover and more computationally intensive to evaluate. For this reason the following three observations have been taken into account during their construction:

1. It is important to note that the covering relationship is purely an optimisation and that Siena would work correctly without it, albeit not as efficiently as with it [8]. Hence, the evaluation of covering relationship as false, when it is actually true, can affect the routing efficiency, but can not affect the routing accuracy. In other words, false negatives reduce routing efficiency, but do not affect routing accuracy.
2. Evaluation of the covering relationship as true, when it is actually false, can affect routing accuracy. In other words, false positives do affect routing accuracy.
3. Absence of false negatives in the covering relationship algorithm would not necessarily result in optimal routing performance. In a dynamic environment, where subscription filters have a short lifespan, a more precise covering relationship algorithm could cause a high computational burden and result in bad overall routing performance.

Considering this, and with a goal to keep things simple, the covering relationships between composite bag operators, which allow for some false negatives, have been defined. They are presented in the table 2. The correctness of these relationships is based upon transitivity for composite bag operators. All symbols used in the table have the same meaning as in the simple bag operators case, with an addition of λ and ε that represent sub-operators of composite bag operators.

A	B	A covers B when	
$x \text{ equals}_\lambda P$	$x \text{ equals}_\varepsilon Q$	$\lambda \equiv \varepsilon$ AND	λ is transitive AND
$x \text{ superbag}_\lambda P$	$x \text{ equals}_\varepsilon Q$	$\lambda \equiv \varepsilon$ AND	λ is transitive AND
$x \text{ subbag}_\lambda P$	$x \text{ equals}_\varepsilon Q$	$\lambda \equiv \varepsilon$ AND	λ is transitive AND
$x \text{ equals}_\lambda P$	$x \text{ superbag}_\varepsilon Q$	Never	
$x \text{ superbag}_\lambda P$	$x \text{ superbag}_\varepsilon Q$	$\lambda \equiv \varepsilon$ AND	λ is transitive AND
$x \text{ subbag}_\lambda P$	$x \text{ superbag}_\varepsilon Q$	Never	
$x \text{ equals}_\lambda P$	$x \text{ subbag}_\varepsilon Q$	Never	
$x \text{ superbag}_\lambda P$	$x \text{ subbag}_\varepsilon Q$	Never	
$x \text{ subbag}_\lambda P$	$x \text{ subbag}_\varepsilon Q$	$\lambda \equiv \varepsilon$ AND	λ is transitive AND

Table 2: Covering relationships between composite bag operators

It is important to note that the covering relationships of other Siena operators are not affected. Their descriptions are available in [1] and [9], and remain completely unchanged.

3. THE SEMANTIC EXTENSION

Knowledge-based Networking, an extension of Content-based Networking involves routing an event across a network, based not just on the values of an event's contents but also on some semantics of the data and associated meta data contained in the event. We have developed a model for the filtered dissemination of semantically enriched knowledge over a large loosely coupled CBN of distributed heterogeneous agents. We call such a semantic-based CBN a Knowledge-Based Network (KBN). In [10, 11, 12, 13] a KBN implementation is presented that extends Siena by providing three additional ontological base types: properties, concepts/classes and individuals/instances, as described in ontologies originating from the semantic web ontology. It also supports subsumptive subscription operators, i.e. sub-class/property (MORESPEC) i.e. *more specific*, super-class/property (LESSSPEC) i.e. *less specific*, and semantic equivalence (EQUIV). For example, as seen in the Wine ontology [14], the ontological type "wine" is less specific than (subsumes) the type "white wine", or "white wine" is more specific than "wine" since "wine" is a superclass of "white wine". Producers of knowledge express the semantics of their available information based on an ontological representation of that information. Consumers express subscriptions upon that information as simple semantic queries. If an event consumer was interested in receiving events about some ontological entity E, classes equivalent to E, or entities more specific than E, this can be easily achieved by creating a filtering constraint such that the entity described in a field x of the message is subsumed by E, i.e., (x MORESPEC E). E.g., a subscriber can subscribe to all KBN messages that contain an attribute whose value is a concept more/less specific than the named concept in the subscription. This approach provides loose semantic coupling between applications, which is vital as a new wave of applications increasingly rely on using the application information, context and services offered by existing heterogeneous distributed applications. To achieve this, each KBN router holds a copy of a shared OWL ontology, within which each ontological class, property and individual used is described and reasoned upon.

The new ontological types: classes, individuals, and properties; are first class KBN types, and can be used in any KBN subscription or notification, along-side the standard Siena types and operators. This allows messages to be matched to subscriptions based on extensible type information, which can effectively represent meta-data for the message without having to maintain an ever-growing set of universal attribute names. Instead, a simple set of shared attribute names can be used for a concept type, which uses values from a taxonomy that is maintained, distributed and reasoned over at run-time using existing standardised ontology techniques.

In this paper we extend the KBN implementation in [10, 11, 12, 13] to add to the existing semantic operators. The semantic operators previously available were: equivalent class, equivalent property, equivalent individual (EQUIV); subclass, subproperty (MORESPEC); and superclass and superproperty (LESSSPEC). The new operators are: ISA, IS_NOT_A, ONTPROP, and NOT_EQUIV.

3.1 The New Semantic Operators

The MORESPEC, LESSSPEC and EQUIV operators are described in the previous subsection.

The **ISA** operator is used to match an ontological individual/instance against its ontological types/classes. If an individual I is defined as being of an instance of a certain type C then the ISA operator will match the individual I to class C, all classes equivalent to C, and all superclasses of C. For example if the person "John" is represented in an ontology as an individual of the class "person", and the class "person" is defined to be equivalent to the class "human", and the class "human" is a subclass of the class "mammal", then the individual "John" is related by the ISA relationship to the classes "person", "human", and "mammal". So the subscription filter (x ISA "human"), where x is an ontological individual, would match a notification that contains the named attribute (x : "John"). This kind of a subscription filter was not previously possible since the EQUIV, MORESPEC, and LESSSPEC operators could only compare classes with classes, properties with properties, and individuals with individuals.

The **IS_NOT_A** operator is again used to compare an ontological individual/instance against its ontological types. If an individual I is defined as being of an instance of a certain type C then the IS_NOT_A operator will match the individual I to all classes except class C, all classes equivalent to C, and all superclasses of C. Based on the example above the individual "John" is a "human" and is a "mammal", but is not a "cow" where "cow" is a subclass of "mammal". So the subscription filter (x IS_NOT_A "cow") would match a notification that contains the named attribute (x : "John").

The **ONTPROP** operator is used to match ontological individuals against each other using any ontological object property. Ontological object properties define named relationships between individuals of two classes. For example the object property "eats" might be defined between individuals of type "animal" and individuals of type "food", so the individual called "Colleen" of type "cow" ("cow" is a subclass of type "animal") would be related by this "eats" relationship to an individual called "grass" of type "food". So the filter (y ONTPROP "eats" "grass"), where y is an ontological individual, would match a notification that contains the named attribute (y : "Colleen").

Since Siena does not support a generic NOT (!) operator, the operators NOT_EQUIV was added for completeness. This operator is the opposite of the ontological EQUIV operator discussed above. It is used to compare classes with classes, properties with properties, and individuals with individuals.

3.1.1 The Covering Relationships between Semantic Operators

Similar to Composite bag operators the Semantic Operators have complex semantics and for this reason covering relationships between them are harder to discover and more computationally intensive to evaluate. Again, with a goal to keep things simple, the covering relationships between semantic operators, which allow for some false negatives, have been defined. They are presented in the table 3 on the next page. In particular the NOT_EQUIV and ONTPROP operators could have more inclusive covering relationships, for example $ONTPROP_{prop1}$ could cover $ONTPROP_{prop2}$ where $prop1$ is not equivalent to $prop2$ but they are related in some other transitive way.

A	B	A covers B when
x EQUIV a	x EQUIV b	b EQUIV a
x MORESPEC a	x EQUIV b	b MORESPEC a
x LESSSPEC a	x EQUIV b	b LESSSPEC a
x EQUIV a	x MORESPEC b	Never
x MORESPEC a	x MORESPEC b	b MORESPEC a
x LESSSPEC a	x MORESPEC b	Never
x EQUIV a	x LESSSPEC b	Never
x MORESPEC a	x LESSSPEC b	Never
x LESSSPEC a	x LESSSPEC b	b LESSSPEC a
x EQUIV a	x NOT_EQUIV b	Never
x MORESPEC a	x NOT_EQUIV b	Never
x LESSSPEC a	x NOT_EQUIV b	Never
x NOT_EQUIV a	x EQUIV b	Never
x NOT_EQUIV a	x MORESPEC b	Never
x NOT_EQUIV a	x LESSSPEC b	Never
x NOT_EQUIV a	x NOT_EQUIV b	b EQUIV a
x ISA a	x ISA b	b MORESPEC a
x IS_NOT_A a	x ISA b	Never
x ISA a	x IS_NOT_A b	Never
x IS_NOT_A a	x IS_NOT_A b	a MORESPEC b
x ONTPROP _{prop1} a	x ONTPROP _{prop2} b	a EQUIV b AND prop1 EQUIV prop2

Table 3: Covering relationships between semantic operators

4. IMPLEMENTING THE EXTENSIONS

The two extensions discussed in this paper have been fully implemented and tested, and a deployable Java KBN implementation is available.

One of the crucial tasks in the development of Siena bag extension was the implementation of the algorithm for the comparison of bags by the composite bag operator. The algorithm used is a very simple brute force algorithm that clearly illustrates the performance of the composite bag operator. It simply checks all possible arrangements of one bag with another until a matching arrangement is found, or else it returns unsuccessful. As discussed in the section 5, performance, this is not ideal. An optimisation of this algorithm would be to sort the bags (at least partially) before applying the bag comparison to the bag. This has no effect on the contents of the bags, since bags are defined to be unordered, but it can significantly optimise the matching of bags, as shown in Section 5.

Previous works by the authors have shown that semantic types and operators can be supported by incorporating an ontological knowledge base and an ontological reasoner into each Siena KBN router/broker [10, 11, 12, 13]. To achieve this there have been significant additions to the codebase to support the new ontological types and operators. However, it must be noted, the operation of the KBN as a non-semantic Siena CBN has not been compromised in any way. The semantic extensions (and bag extensions) are provided to supplement the Siena CBN system.

Previous works by the authors have also shown that the loading of new ontologies into a reasoner embedded in a KBN node is computationally expensive due to load-time inference [11]. Therefore the frequency of changes to the ontological base of a given KBN must be minimised since changes will need to be distributed to each of the nodes in the network. Secondly, ontological reasoning is memory intensive and memory usage is proportional to the number of concepts and relationships loaded into the reasoner so reasoning latency can be controlled by limiting this number in any given KBN node. However, once loaded and reasoned over, the querying of such an ontological base is relatively efficient, with performance relative to size of the ontological base [11].

A large number of ontology reasoners are available, including: KAON2 [15], Pellet [16], Racer [17], FaCT [18], F-OWL [19]. Any choice of a reasoner must be based on examination of performance evaluations in the literature, such as [20, 21, 22, 15], as well as with separate benchmarking. These evaluations must also be compared to the performance characteristics of domain specific reasoners, or existing reasoners cut-down to give reduced but sufficient results in return for enhanced performance. This tradeoff of reasoning performance versus expressiveness and accuracy of the model after the inference cycle is of particular importance where such reasoning may be required for efficient and correct routing in the network. The differing performance characteristics of different reasoners under different conditions, such as, the impact of the ratio of concepts to relationships or of subsumption relationships to user defined predicates, must also be evaluated. The performance of different reasoners, and the reasoning load, will also change in a non-linear fashion depending on the size and expressiveness of the ontologies used and the level of ontology language used (e.g. OWL-Lite vs. OWL-DL) [20, 21, 23, 22, 15]. Of particular importance is the amount of reasoning that can be performed at ontology load time versus when the first or subsequent queries are submitted to the ontology. This becomes particularly important if ontologies are added or removed dynamically, as would be typical in a network of knowledge producers and consumers where joins and departs from the network will occur. For this paper the Pellet ontology reasoner [16] is used with Jena [24].

5. PERFORMANCE

In evaluating a KBN's performance we focus less on the performance of the underlying network and more on the performance of individual routers. The KBN is an overlay network, the aim of this research is not to optimise its network level performance, this is an issue which would need to be addressed by Siena's authors. The aim of this work is to address the routers ability to perform the calculations required for matching the subscriptions to publications. For example taking the Hermes [25] CBN it would be desirable to transfer the end result of this work to operate on a different routing technology. This does not mean that we disregard the applications effect on its underlying network, we will however focus our evaluation the performance of the individual router. It is the authors' opinion that these evaluations best report the performance of the KBN as a contribution to the field of event based systems in contrast to an evaluation of the Siena CBN routing mechanism on network load.

5.1 Calculating the Overhead of the Bag Extension

The evaluation of the bag extension for Siena requires verification of the correct behaviour of the system and evaluation of the system performance. The correctness of the system behaviour has been empirically confirmed by applying human performed functional test cases. The performance evaluation should investigate the following:

- performance of notifications containing bags
- subscription/routing tree lookup performance for notification matching where notifications and subscriptions contain bags
- overhead of merging subscriptions containing bag into the subscription/routing tree

5.1.1 Performance of Notifications Containing Bags

Performance of notifications containing bags has not been specifically addressed, since a bag of the size n is roughly equivalent to n attributes with the scalar values equal to the bag's elements. Actually the size of the equivalent bag attribute is even smaller since it contains only one attribute name compared to n attribute names of the equivalent scalar attributes. Therefore the performance of a notification containing a bag attribute is expected to be at least as good, and probably better, than the performance of a notification containing equivalent set of scalar attributes.

5.1.2 Performance of Subscription Tree Merge and Lookup using Simple Bag Operators

The performance of subscription filters utilising simple bag operators was also not specifically addressed, since the algorithm for comparing bags with simple bag operators is straightforward. Suppose there exists some bags B and C . Without loss of generality we can assume that $|B| \leq |C|$. The simple bag operator comparison has the best time complexity $O(|B|)$ and the worst time complexity $O(|C|^2)$. If bags are pre-ordered, assuming bag elements can be totally ordered, it is possible to use even better algorithm that has the worst time complexity of $O(|C|)$.

5.1.3 Performance of Subscription Tree Merge and Lookup using Composite Bag Operators

With the brute force algorithm used to compare bags with the composite bag operator, in the most optimistic case the algorithm finds matching elements immediately. In this case the time complexity of the algorithm is $O(|B|)$ for some bags B and C where $|B| \leq |C|$.

In the most pessimistic case the algorithm must fully match all possible arrangements of the elements of the smaller bag with the elements of the bigger set. In this case the time complexity is:

$$O\left(\frac{|C|!}{(|C|-|B|)!}\right) < O(|C|^{|B|})$$

We can see from the formulas above that the time complexity depends above all on the size of the smaller bag, because the size of the smaller bag appears in the exponent of the time complexity. It is evident that the algorithm for composite bag operators implemented in the scope of this research is truly useful only for bag comparisons where at least one of the bags is small.

It is possible to develop much more effective comparison algorithms for certain specialized composite bag operators. For

example, it was suggested to exploit the ordering of the elements to improve the algorithm.

For composite bag operators over integer bags, where the sub-operator is one of the $<$, $\leq$, $>$, $\geq$ it is possible to develop a very effective comparison algorithm if bags are pre-ordered. This is because the set of all integers is a totally ordered set with regard to $\leq$, or $\geq$. Such an algorithm would have a very low time complexity of $O(\max(|B|, |C|))$. Unfortunately not all sets are totally ordered. For example, ontological concepts form only a partially ordered set with regard to subsumes. An effective algorithm for composite bag operators over partially ordered sets could be a subject of the future research. At this stage it is obvious that the overhead involved in at least partially sorting the bags before they are compared is of benefit. Alternatively, there might exist some constraints that narrow the set of all possible bags that are to be published in notifications or subscription filters, to some subset, for which good performance of the composite bag operator could be guaranteed.

5.2 Measuring the Performance of the KBN

One of the main questions that surround the use of ontologies deep in the network at the routing layer remains the evaluation of the resulting performance overhead. Previous small scale studies in this area [11, 26, 12] show a definite performance penalty, but this may be acceptable when offset against the increased flexibility and expressiveness of the KBN subscription mechanism. For the purposes of comparison we carried out a further evaluation of the time taken to merge a subscription into a KBN router, and the time taken to match a publication against a router's subscription table. As discussed we are not concerned in this paper with the network overhead of our KBN implementation, since almost completely determined by the Java version of Siena we are using [5]. For this reason our test bed was very simple, one KBN router (*siena.HierarchicalDispatcher, extended*) and one client (*siena.ThinClient*), connected to the router over the localhost loopback network interface (*127.0.0.1*). This forces all traffic to be marshalled between the client and the server, but not sent out on the network.¹

Figures 1 and 2 together show the first experiment, the running average of the time taken for a KBN router to merge a subscription into its subscription table. For each type of subscription, we measured the time to merge 100 dynamically subscriptions into the subscription table. Between each set of subscriptions the subscription table was cleared to avoid interference. Each subscription had only one filter. The 7 types of subscriptions we used were:

- That named attribute x should be a specified integer, where the specified integer was drawn randomly for the set of integers 0-1000
($x = \text{random integer } 0-1000$)
- That named attribute x should be a specified string, where the specified string was drawn randomly from the dictionary of 100 random English nouns
($x = \text{random string}$)

¹ All tests were carried out a single moderately loaded development workstation: Dell Dimension, 9200, Intel Core 2 Duo 2.66 GHz processor, 4GB ram, Windows XPx64. Java 1.6.0_01 (64 bit). Jena version 2.4 to handle ontologies, with the Pellet reasoner version 1.5.1

- That named attribute x should be a substring of a specified string, where the specified string was drawn randomly from the same dictionary of 100 random English nouns
(x *SUBSTRING random string*)
- That named attribute x should be a subbag, i.e. $\subseteq$, of a specified bag, where the specified bag contained 10 integers drawn randomly for the set of integers 0-10
(x *SUBBAG 10 random integers*)
- That named attribute x should be a superbag when the substring suboperator is applied, i.e. $\supseteq_{\text{SUBSTRING}}$, of a specified bag, where the specified bag contained 5 strings drawn randomly from the same dictionary of 100 random English nouns
(x *SUPERBAG+substring 5 random strings*)
- That named attribute x should be an Ontological class which is a subclass or equivalent to, i.e. $\supseteq_{\text{MORESPEC}}$, the specified class where the specified Ontological class is one of the 137 classes specified in the Wine ontology [14].
(x *MORESPEC random ontology class*)
- That named attribute x should be an Ontological individual which is related to a specified individual according to some specified Ontological object property/relation, where the individual is one of the 208 individuals, and the property is one of the 16 object properties specified in the Wine ontology [14]
(x *ONTPROPERTY random ontology individual*)

The Siena subscription table is implemented in such a way that for the subscription to be added is repeatedly compared to and evaluated against the other subscriptions in the subscription table in order to find the appropriate place to insert the subscription. As can be seen from most of the plots in figures 1 and 2 the time taken to merge a subscription is mostly linearly dependent on the number of subscriptions already in the subscription table. This is because of the unlikelihood of the newly arriving subscription being similar enough to an already received subscription to allow the new subscription to be aggregated with the previous subscription. This however is not the case with the MOREPSEC subscription, where the time to merge a new subscription grows sub-linearly with the number of subscriptions present. This is partly due to the size of the Wine ontology, but mostly due to the fact that the ontology's classes have been reasoned into a hierarchy of classes.

Except for the ONTPROPERTY subscriptions, which are significantly more complex than the other subscriptions, the time taken to merge a subscription was less than 2 milliseconds. This justifies our claim that we should only measure the overhead inside the router rather than include network overhead, which would substantially obscure the findings of our study.

The bags used in this evaluation have integer or string elements, which are compared with the number and string sub-operators. Alternatively, bags over ontological values could be used together with an ontological suboperator. This would increase all measured delays approximately by a constant factor, but it would not affect the ratios between the measured values.

In addition to evaluating the overhead of merging a new subscription into the KBN router's subscription table we also measured the time to match notifications against the subscription table. For this experiment we re-submitted all of the 700 subscriptions that arose from the previous experiment, i.e. 100 dynamically generated subscriptions for each of the 7 types of

Figure 1: Time to merge subscriptions (1)

Figure 2: Time to merge subscriptions (2)

subscriptions discussed above. We then synthetically generated sets of subscriptions to perhaps match against those subscriptions. For each of 6 types of notifications, 100 notifications were published to KBN router. The 6 types of notification we used were:

- That named attribute x had a random integer value between 0 and 1000
(*A random int. 0-1000*)
- That named attribute x had a string value drawn randomly from the same dictionary of 100 English nouns used in the subscriptions
(*One of 100 random strings*)
- That named attribute x had a bag of 5 integers, where the integers were randomly selected from the range 0 to 10
(*A bag of 5 ints 0-10*)
- That named attribute x had a bag of 10 strings, where the string values were drawn randomly from the same dictionary of 100 English nouns
(*A bag of 10 strings, out of a set of 100 strings*)

- That named attribute x was an Ontological class, where the Ontological class was one of the same 137 classes specified in the Wine ontology [14] as used in the subscriptions.
(One of 137 Ontological classes)
- That named attribute x was an Ontological individual, where the Ontological individuals was one of the same 208 individuals specified in the Wine ontology [14] as used in the subscriptions.
(One of 208 Ontological individuals)

For each publication types table 4 shows the average time taken to match that type of subscription against the entire populated subscription table with 700 subscriptions, the standard deviation of match times and the total number of matches that actually accrued for the 100 publications. As can be seen from table 4, the time taken to match a publication against the total set of subscriptions is still a relatively inexpensive operation. However, the times taken to perform the ontological matches are relatively slower, but still within acceptable ranges. In these evaluations we have shown that the runtime overheads of applying our extensions are very small by comparison to the substantial improvement in subscription flexibility and expressiveness.

Value for x in the notification	Subscription match time (ms)	Std. Dev.	Num. of matches
A random int. 0-1000	0.98	0.57	8
One of 100 random strings	2.04	0.99	201
A bag of 5 ints 0-10	2.18	1.31	543
A bag of 10 strings, out of a set of 100 strings	1.64	0.55	0
One of 137 Ontological classes	15.62	1.71	335
One of 208 Ontological individuals	29.28	3.58	1

Table 4: Performance of subscription matching

6. MOTIVATIONAL CASE STUDIES

The advantages of using ontologies have been argued extensively, but the main advantage is that ontologies attempt to capture the precise meaning of terms with respect to other terms. Furthermore, ontologies can be used for reasoning and inference (for example, consistency checking or drawing conclusions from knowledge contained in the ontology but not necessarily encoded explicitly). Bags meanwhile allow much more flexible subscription filters. In this section we present a number of case studies that show why these extensions are useful. The case studies are intended to motivate these extensions, full treatment of these case studies are either outside of the scope of this paper or are discussed elsewhere.

6.1 Decentralised Semantic Service Discovery

A key advantage of using a service-oriented architecture via Web services is the ability to compose services from a number of constituent services operated by different organisations. Currently, however, the runtime discovery of such compositions is centralised. A typical Web service architecture consists of: service providers that create and publish Web services, service brokers that maintain a registry of published services and support their

discovery, and service consumers that search the service broker's registries. The discovery of individual Web services that can act as a service specified within a composition typically relies on searching centralised repositories recording Web service offering.

The extended KBN described in this paper has been used to support dynamic and decentralised semantic service discovery [7]. The system exploits the use of ontology-based descriptions of Web services in OWL-S [27] to achieve effective discovery and loose coupling. We describe participating processes in abstract terms of required capabilities, which describe the process solely in terms of inputs, outputs, preconditions and effects. Part of the complicated process of composing a composite service for constituent services is this matching of outputs of one service to inputs of the next service, and matching the effects of one service to the preconditions of the next service. All in such a way the combined composite service performs the task required, given available inputs and provides required outputs. Although the entire process of not just discovering and orchestrating services, but also controlling and choreographing their execution in a completely decentralised manner is explained in detail in [7], let us focus here on matching semantic service inputs and semantic service outputs.

Services announce themselves with a KBN notification, which includes (among other parameters) a description of its inputs expressed as a *bag* of semantic classes, and a *bag* of outputs expressed as semantic classes. When a service is required, a KBN subscription is created. This subscription can then use composite bag operators to find compatible services or subservices. The KBN middleware then routes service notifications from service providers to service consumers, thereby performing the discovery filtering within the network. If the bag of inputs required by an available service is a sub-bag of the bag of available inputs when the superclass suboperator is applied ($ServiceInputs \subseteq_{LESSSPEC} AvailableInputs$), and the bag of outputs from an available service is a super-bag of the bag of the required outputs when the subclass operator is applied ($ServiceOutputs \supseteq_{MORESPEC} RequiredOutputs$), then the available service's interface is appropriate. If this is encoded as a KBN subscription, allowing for any additional subscription filters to be also included in the subscription, then the KBN can act as a decentralised service discovery platform. For more details on how the other aspects of discovering, orchestrating and choreographing services is achieved, please refer to [7].

6.2 News Feed Distribution and Subscription

Users of the web are increasingly interested in tracking the appearance of new postings on the web rather than locating existing knowledge. The time, at which, information items on the internet are posted is increasing in importance relative to the content of the post, e.g. blog postings rapidly fade in importance as time passes. The web has responded to this need with RSS feeds which allow event postings to be quickly notified to interested users. News or other event feeds have emerged as an important component of the Web 2.0 movement. However, this system relies on users subscribing to feeds of pages they have already located, whilst feed aggregators offer only rudimentary searches or simple classifications of feeds. This is partly because the near-real time events present in feeds are disassociated from the system of user-defined hyperlinks required by search engines which also introduces a discovery latency that is unacceptable to

feed consumers. Current event-based publish-subscribe systems offer a networking model that is well suited to such applications, and a few global-scale implementations have emerged supporting high value applications such as stock feeds. These examples are typically limited to relatively static characterisations of events and there have been few examples of applying Semantic Web techniques to the efficient distribution of events.

Having established an indication of the most popular RSS feed types we decided to study some real-world distributions of publications and subscriptions for news feeds [13]. We found that news feeds and their publications are typically marked up with some degree of metadata. Among other information, this metadata typically includes some classification or category information of what the feed and notification is about, typically containing a set of keywords. We found that the category information was usually drawn from some taxonomy of categories, and this taxonomy could be easily translated into an ontology of semantic classes.

In this way, notifications of new, news feed events, could be codified as a KBN notification containing the URL of the feed the message, some information about the author etc, an attribute containing a bag of ontological classes as the subject categories, and a bag of keywords. Consumers of event could then receive events based on a KBN subscription. Among other filters the subscriber would specify a bag of zero or more required keywords and a bag of zero or more categories. Firstly the bag of keywords in the event notification should be a simple superbag of the keywords in the subscription ($EventKeywords \supseteq RequiredKeywords$). If the subscription contained a bag of *required* categories then the bag categories in the event notification should be a *superbag* of the bag of categories requested or equivalent. If the subscription contained a bag of *suggested* categories then the bag categories in the event notification should be a *subbag* of the bag of categories requested or equivalent. Since the categories were arranged taxonomically, the subscription should match equivalent categories and their sub-categories, so the subscription used the MORESPEC suboperator ($EventCategories \supseteq_{MORESPEC} RequiredKeywords$) or ($EventKeywords \subseteq_{MORESPEC} SuggestedKeywords$). For more details on this scenario, and a detailed evaluation of the performance of the KBN in this usage scenario, refer to [13].

6.3 Distributed Correlation of Faults in a Managed Network

Increasingly there is a demand for more scalable fault management schemes to cope with the ever increasing growth and complexity of modern networks. However, traditional fault management approaches typically involve rigid and inflexible hierarchical manager/agent topologies and rely upon significant human analysis and intervention, both of which exhibit difficulties as scalability and complexity increases. Our distributed correlation scheme [28, 29] distributes correlation tasks amongst an entire network of fault agents, where each agent takes a role in part of the correlation. These distributed agents can be arranged so that low level correlators provide sub correlation results for higher level correlation agents, and the whole correlation task for the managed network can then be performed hierarchically.

Event information, correlation rules and the event correlation graphs are all represented in this scheme as ontologies. The use of an ontology representation not only enables these elements to be

easily changed, but also (through reasoning) provides an opportunity for self configuration of the fault correlation system itself to be done automatically in reaction to context changes. We have also published numerous works describing the benefits of using semantic mark-up in network fault management [26, 12, 30]

In one part of the work described in [28, 29] we arranged high-level and low level events in a hierarchical manner according to a “caused by” relationship, where low level events cause high-level events. This was then codified using the ontological subclass/superclass relationship. A correlation agent could then subscribe to all events at a certain level or all events that could cause that event, using the semantic MORESPEC operator. If the agent was interested in a combination of events, then it could subscribe to a flexible bag of causing events that may have occurred together. Once an agent discovered or calculated a correlation it would announce this as a higher-level event, using a KBN notification. (By including a bag of information about what triggered this correlation a top level agent could then perform root-cause analysis of what caused a top-level fault!).

However, we found that this mapping of caused-by relationships may not be easily map-able to a subclass/superclass relationship, and the use of this relationship to codify a caused-by relationship was breaking the semantics of the concept hierarchy. It was this, combined with several other reasons, that prompted us to develop the generic ONTPROPERTY operator, where the caused-by relationship, and similar relationships, could be codified directly as ontological object properties without rearranging the natural hierarchy of event types. Therefore agent subscription(s), would then match interesting events according to this causes/caused-by ontological property, ($FaultInstance \text{ ONTPROPERTY}_{CAUSED\text{BY}} SubFault$). This could then be easily expanded to make use of the bag extension.

There is neither tight coupling between the network of managed elements or specific correlation agents due to the usage of the semantic publish/subscribe middleware. All events, including raw fault events are pushed into the fault correlation network, but if no agent is interested in that event then the event is quenched immediately. If an agent is interested in the event then it is routed to that agent. If there are no events in the network the correlation agent takes up minimal resources. In addition, a failure in one specific correlation agent can not disable the whole fault management system, as another correlation agent can assume the correlation task of the failed agent by adjusting its subscription.

6.4 Context Distribution

Pervasive computing promises to make available a vast volume of context messages from environmental sensors embedded in the fabric of everyday life reporting on user location, sound levels and temperature changes, to name but a few. Any scalable context delivery system must ensure therefore the accurate delivery of context events to the consumers that require them. However, the wide range of sensors and sensed information, and the mobility of consuming clients, will present a level of heterogeneity that prevents consumers accurately forming queries to match possibly unknown forms of relevant context events. As context-aware systems become more widespread and more mobile there is an increasing need for a common distributed event platform for gathering context information and delivering to context-aware applications. However, most Pub-Sub systems require agreements on message types between the developers of producer and

consumer applications. This places severe restrictions on the heterogeneity and dynamism of client applications. Here we see an ideal application potential of Knowledge-based Networks for the filtered dissemination of context over a large loosely coupled network of distributed heterogeneous agents, while removing the need to bind explicitly to all of the potential sources of that context. The likely heterogeneity across the body of context information can be addressed using runtime reasoning over ontology-based context models. A KBN based on semantically enhanced messages and corresponding expressive and flexible queries is far more flexible, open and reusable to new applications. For this reason we foresee the application of, and have already applied, the KBN in numerous context-aware scenarios [12, 47, 26].

7. RELATED WORK

There has been little examination of the use of ontology-based semantics in content-based networking in scientific literature.

The rationale behind semantics within messages flowing across CBNs is supported through the work of Baldoni et al [31] in which each publication within the network is tagged with a topic. This can be likened to assigning to an email message a subject line, except in the case of [31] the message is assigned a topic tag. The subscription table in [31] is constructed using a list of couples $\langle t, i \rangle$ where t is the topic a node is subscribed to and i is the corresponding topic overlay identifier. Upon retrieving a new subscription for a topic the subscription management component adds an entry for the topic to the subscription table and then passes the task of connecting the corresponding overlay networks. In the tagging of messages it is easy to see that this increased level of descriptiveness is beneficial in the task of matching messages. As described earlier in this paper a collection of tags can be formed together to create a bag. In [13] bags are used to store keywords, or tags, associated with the message being sent, in fact forming a key part of the message itself. This allows for the subscription to a sub/superset of a bag of keywords. In a set of keywords, or tags, strings represent the knowledge and only through interpretation is a relationship mapped between tags. For example the tag string "DEBS2008" has no semantic relationship to the tag string "DEBS2007". However if the tag was changed to an ontological type then DEBS2008 and DEBS2007 could be described either as subclasses or instances of an ontological concept called "DEBS".

In [32] a semantic publish-subscribe system is presented, but it is based on a centralised (pub-sub bus) implementation and thus is limited to enterprise scale and does not offer true CBN capabilities. In [32] three approaches to enhance subscriptions and events semantically were proposed in order to make the existing centralised syntactic matching algorithm semantic-aware while keeping the efficiency of current event matching technique. However, [32] does not address the scalability issues involved in including ontology-based reasoning into the CBN, and no proposal is made to integrate this with the P2P routing extension for ToPSS. More significantly, however, no report of an implementation or evaluation of this proposal has yet emerged.

Another ontological pub/sub system called Ontology-based Pub/Sub (OPS) system is developed by [33]. Aiming to improve expressiveness of events and subscriptions, it uses RDF and DAML+OIL techniques to describe events and subscriptions, where events and subscriptions are represented as RDF graphs

and graph patterns respectively. The application concepts within events are integrated together to form a concept model that is represented as ontology, so OPS can match events with subscriptions both semantically and syntactically. Meanwhile, the authors designed a highly efficient matching algorithm in order to improve the scalability of system. The main idea of the OPS matching algorithm is that it builds an index structure (according to concept model) of possible statement patterns (decomposed by RDF graph pattern) that are the basic units of matching, and uses AND-OR trees as matching trees to avoid the backtracking of RDF graph. However, this system does not include the ability to also perform generic content-based subscriptions so it is not as expressive or flexible as the KBN system presented here.

A further KBN network described in [34] support ontological publications and subscriptions. This system uses a SPARQL as its subscription query language. Again, this system does not include nor have the ability to perform generic content-based subscriptions. In addition the substantial overhead of using SPARQL, and the lack of a mechanism to aggregate subscriptions, means that the overhead and scalability of this system is unclear.

In [35], semantics can be used in messages in a pub-sub middleware; however, the semantics are used only at the edge of the network in a manner similar to a small scale study presented by the authors of this paper in [26]. The KBN presented within this paper uses semantics deep in the forwarding algorithm of each message router within the network.

There have been several attempts at applying P2P DHT techniques to the retrieval of distributed ontology encoded knowledge information, e.g. in RDF, in semantic overlay networks [36, 37, 38]. However, these systems are focused on query-response communication, rather than the Pub-Sub model.

As surveyed in [39] and [40] there are many type-based and topic/subject-based distributed event based systems. Many of these systems offer a form of hierarchical addressing, which permits programmers to organize topics according to hierarchical containment relationships. These mechanisms allow subscribers to use the notion of subclass and superclass and are therefore similar to some parts of the implemented semantic extension described in this paper. However, these systems, especially the topic/subject-based systems do not provide the same level of expressiveness and flexibly.

In XML-based systems, for example [41, 42, 43], events are published as XML documents. Subscriptions are in the form of XPath [44] expressions. The list of queries would be applied to a DOM tree created from the XML document and the message would be dispersed to a broker when a match is found. While XML-based systems provide more expressiveness than topic-based publish-subscribe, the message architecture is based on tree patterns and so supports less expressiveness and flexibility than advocated by the authors of this paper.

A large number content-based pub-sub systems support disjunction type subscripsts and the use of sets in their subscriptions. We argue that our work on the bag extension to Siena now allows Siena to support such queries. However we argue that the composite bag operators supported by our KBN surpasses this with the ability to define even more expressive and flexible subscriptions, especially when the composite bag operators (unique to this work) are use, particularly when used together with the ontological types and operators.

8. DISCUSSION AND FURTHER WORK

This paper defines and discusses the implementation of two novel extensions to the Siena Content-based Network to extend it to be a Knowledge-based Network (KBN). One extension provides ontological concepts as an additional message attribute type, onto which subsumption relationships, equivalence, type queries and arbitrary ontological relationships can be applied. The second extension provides for a bag type to be used that allows bag equivalence, sub-bag and super-bag relationships to be used in subscription filters, composed with any of the Siena subscription operators or the ontological operators previously mentioned. The performance of this KBN implementation has also been explored. This research has only just begun to explore applications for the expressiveness of the knowledge-based networking. As presented in the motivational case studies above, ongoing research by the authors is focussing on how our KBN implementation can be applied across a wide selection of application areas:

- decentralised semantic service discovery [7]
- discovery and change notification of policies between federated communication service providers
- sensor readings in a multi-domain heterogeneous ubiquitous computing application
- RSS with semantic markup in Web 2.0/Semantic Web [13]
- semantically rich notifications from heterogeneous network elements in OSS [26, 12, 30]
- distributed fault correlation using semantically rich notifications [28, 29]
- semantically rich notifications about changes in financial markets
- semantic routing of multimedia (MPEG) stream with semantic meta-data

Ongoing work is also focussing on extending the Knowledge-Based Network to incorporate semantic-based clustering, this work aims to provide a network environment in which routing nodes, publishers and subscribers are clustered based on their semantic footprint and interests. The benefits of this are threefold: Firstly, this reduces the processing time involved in making routing decisions based on messages content. It takes fewer hops to get from source to destination, as these are already closely linked based on the likelihood of there being a match between the two. Secondly, this allows for natural grouping of likeminded publishers and subscribers as seen in traditional web forums / newsgroups. Thirdly, it allows certain area of the network to have specialised sub/super ontologies which do not need to contain the semantics of the whole network, which means that knowledge base sizes can be reduced and knowledge base updates can be localised. The cluster-based approach to pub/sub networks turns the normal user-based search paradigm full circle as network data is passed from node to node towards those who are most likely to be interested in the data as opposed to those users searching out that same data. In our initial work clusters were statically designed and operated [13]. In this sense nodes are assigned to clusters without the possibility of changing clusters once they have joined, in a manner similar to the approach taken in [31]. This initial clustering method demonstrated how even inflexible and static clustering can have a substantial positive effect. However, we expect that any practical system will need to adapt its clustering to reflect the constantly changing profile of semantics being sent and subscribed to via the KBN, thus creating a network environment

in which messages are passed from node-to-node, cluster-to-cluster based not on the data's destination but based on the message's semantic data. Current work is focussed on allowing users and brokers to join and leave clusters dynamically and independently. Clusters will then be seen as organic structures in which users and brokers join and leave as their own personal interests drift, grow, reform and are refined. Current work is also focusing on integrating policy-based cluster management for the KBN to support much more sophisticated cluster schemes. This will support overlapping clusters and hierarchies of clusters under separate administrative control [11]. In addition, the effect of semantic interoperability in node matching functions and in inter-cluster communications is being assessed [45, 46]. This requires evaluation of different schemes for injecting newly discovered semantic interoperability mappings into the ontological corpus held by KBN routers, as well as how these mappings are shared between routers. Work is ongoing to build on these initial evaluations [45, 46] to design and implement a flexible mapping strategy management framework. Work is also ongoing to investigate how mappings can be dynamically distributed around the network as the knowledge bases of clients joining and leaving the network affect the spread of knowledge across the network. It is foreseen that the KBN itself would be ideal for such a distribution mechanism.

REFERENCES

- [1] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. (2001). Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3).
- [2] Segall, B., Arnold, D., Boot, J., Henderson, M., Phelps, T., "Content-Based Routing in Elvin4", in *proc. AUUG2K*, Canberra 2000.
- [3] Pietzuch, P., Bacon, J., "Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware". *Distributed Event-Based Systems (DEBS'03)*. At *ACM SIGMOD/PODS Conference*, San Diego, CA, June 2003
- [4] Muhl, G., Fiege, F., Pietzuch, P., "Distributed Event-Based Systems". Springer-Verlag, 2006.
- [5] Carzaniga, A., "Siena - Software", <http://www.inf.unisi.ch/carzaniga/siena/software/index.html>
- [6] Weisstein, E. W. (2002). "Multiset. MathWorld – Wolfram Resource". <http://mathworld.wolfram.com/Multiset.html>
- [7] Roblek, D. "Decentralized Discovery and Execution for Composite Semantic Web Services", M.Sc. Thesis, Computer Science, Trinity College Dublin, Ireland, 2006.
- [8] Heimbigner, D., "Extending the Siena Publish/Subscribe Type System," *Department of Computer Science Technical Report CU-CS-946-03*, University of Colorado, 2003.
- [9] Rutherford, M. J. "Siena Simplification Library Documentation 1.1.4. University of Colorado – Web Resource". <http://serl.cs.colorado.edu/carzanig/siena/forwarding/ssimp/namespacesiena.html>
- [10] Lynch, D., Keeney, J., Lewis, D., O'Sullivan, D., "A Proactive Approach to Semantically Oriented Service Discovery". *Innovations in Web Infrastructure (IWI 2006)*. at *World-Wide Web Conf.*, Edinburgh, Scotland. May 2006.
- [11] Lewis, D., Keeney, J., O'Sullivan, D., Guo, S., "Towards a Managed Extensible Control Plane for Knowledge-Based Networking", *Distributed Systems: Operations and Management Large Scale Management*, (DSOM 2006), at *Manweek 2006*, Dublin, Ireland, 23-25 October 2006

- [12] Keeney, J., Lewis, D., O'Sullivan, D., "Ontological Semantics for Distributing Contextual Knowledge in Highly Distributed Autonomic Systems", *Journal of Network and System Management*, Vol 15, March 2007
- [13] Keeney, J., Jones, D., Roblek, D., Lewis, D., O'Sullivan, D., "Knowledge-based Semantic Clustering," in *proc ACM Symposium on Applied Computing*, Fortaleza, Brazil, 2008.
- [14] W3C: "The Wine Ontology", <http://www.w3.org/TR/owl-guide/wine.rdf>
- [15] Motik, B., Sattler, U., "Practical DL Reasoning over Large ABoxes with KAON2", available at <http://kaon2.semanticweb.org/> (2006)
- [16] Parsia, B., Sirin, E. 2004., "Pellet: An OWL-DL Reasoner", Poster at ISWC 2004, Hiroshima, Japan, 2004.
- [17] Haarslev, V., Moller, R. 2001. "RACER System Description", in *proc IJCAR 2001*, volume 2083 of LNAI, 701–706. Siena, Italy, Springer.
- [18] Tsarkov, D., Horrocks, I. "Ordering Heuristics for Description Logic Reasoning", 2005. In *Proc. IJCAI 2005*, 609–614. Edinburgh, UK: Morgan Kaufmann Publishers.
- [19] Zou Y., Finin T., Chen H., "F-OWL: an Inference Engine for the Semantic Web", in *proc Workshop on Formal Approaches to Agent Based Systems*, April 2004, MD, USA, LNCS 3228
- [20] Pan Z., "Benchmarking DL Reasoners Using Realistic Ontologies", in *proc Intl workshop on OWL: Experience and Directions (OWL-ED2005)*. Galway, Ireland. 2005
- [21] Guo Y., Heflin J., Pan Z., "An Evaluation of Knowledge Base Systems for Large OWL Datasets", Technical Report, CSE department, Leigh University, 2004.
- [22] "Pellet Performance", <http://www.mindswap.org/2003/pellet/performance.shtml>
- [23] Guo Y., Heflin J., "LUBM: A Benchmark for OWL Knowledge Base Systems", *Journal of Web Semantics*, Vol. 3 Issue 2., 2005.
- [24] Carroll, J., Dickinson, I., Dollin, C., "Jena: Implementing the Semantic Web Recommendations", in *proc World Wide Web Conference 2004*, 17-22 May 2004, New York, NY, USA. <http://jena.sourceforge.net/>.
- [25] Pietzuch, P., Bacon, J., "Hermes: A Distributed Event-Based Middleware Architecture," in *proc International Conference on Distributed Computing Systems*, 2002.
- [26] Keeney, J., Lewis, D., O'Sullivan, D., Roelens, A., Wade, V., Boran, A., Richardson, R., "Runtime Semantic Interoperability for Gathering Ontology-based Network Context", *Network Operations and Management Symposium (NOMS 2006)*, Canada. April 2006.
- [27] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K., "OWL-S: Semantic Markup for Web Services", W3C Member Submission 22 November 2004
- [28] Wei, T., O'Sullivan, D., Keeney, J., "Distributed Fault Correlation Scheme using a Semantic Publish/Subscribe system," in *proc Network Operations and Management Symposium (NOMS 2008)*, Salvador, Brazil, April 2008.
- [29] Wei, T., "Fault Management System using Semantic Publish/Subscribe approach", M.Sc. Thesis, Computer Science, Trinity College Dublin, Ireland, December 2007.
- [30] Lewis, D., O'Sullivan, D., Power, R., Keeney, J., "Semantic Interoperability for an Autonomic Knowledge Delivery Service", in *proc Workshop on Autonomic Communication (WAC 2005)*, Vouliagmeni, Athens, Greece. October 2005.
- [31] Baldoni, R., Beraldi, R., Quema, V., Querzoni, L., Tucci-Piergiovanni, S., "TERA: topic-based event routing for peer-to-peer architectures," in *proc Distributed event-based systems, (DEBS2007)* New York, NY, USA, 2007.
- [32] Li, H., Jiang, G., "Semantic Message Oriented Middleware for Publish/Subscribe Networks", in *proc of SPIE*, Volume 5403, pp 124-133, 2004
- [33] Wang, J., Jin, B., Li, J., "An ontology-based publish/subscribe system". in *proc ACM/IFIP/USENIX International Conference on Middleware*, 2004.
- [34] Skovronski, J., Chiu, K., "Ontology Based Publish Subscribe Framework". in *proc International Conference on Information Integration and Web-based Applications Services*, 4-6 December 2006, Yogyakarta, Indonesia.
- [35] Cilia, M., Bornhövd, C., Buchmann, A. P., "CREAM: An Infrastructure for Distributed, Heterogeneous Event-Based Applications". *CoopIS 2003*, Catania, Sicily, Italy.
- [36] Tempich, C., Staab, S., Wranik, A., "REMINDIN': semantic query routing in peer-to-peer networks based on social metaphors" *International World Wide Web Conference (WWW)*, New York, USA, 2004.
- [37] Cai, M., Frank, M., "RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network", in *proc of WWW conference*, May 2004, New York, USA.
- [38] Loser, A., Naumann, F., Siberski, W., Nejd, W., Thaden, U., "Semantic overlay clusters within super-peer networks", in *proc Workshop on Databases, Information Systems and Peer-to-Peer Computing in Conjunction with the VLDB 03*
- [39] Meier, R., Cahill, V., "Taxonomy of Distributed Event-Based Programming Systems", *The Computer Journal*, vol 48, no 5, pp 602-626, 2005
- [40] Eugster, P., Felber, P., Kenmarrec, A.M., and Guerrou, R., "The many faces of publish/subscribe". *ACM Computing Surveys (CSUR)*, Vol. 35, Issue 2, (June 2003), 2003.
- [41] Diao, Y., Altinel, M., Franklin, M.J., Zhang, H., Fischer, P., "Path sharing and predicate evaluation for high-performance xml filtering". *ACM Transactions on Database Systems (TODS)*, 28(4):467–516, 2003.
- [42] Gupta A., Suciu, D., "Stream processing of xpath queries with predicates". in *proc 2003 ACM SIGMOD Intl conference on Management of data* pages 419–430, 2003.
- [43] Chan, C.Y., Ni, Y., "Content-based Dissemination of Fragmented XML Data". in *proc International Conference on Distributed Computing Systems*, ICDCS 2006: 44
- [44] Clark J., DeRose, S., "Xml path language (xpath)". <http://www.w3.org/TR/xpath>.
- [45] Guo, S., Keeney, J., O'Sullivan, D., Lewis, D., "Coping with Diverse Semantic Models when Routing Ubiquitous Computing Information", *The Workshop on Managing Ubiquitous Communications and Services (MUCS2008)* at NOMS 2008, Salvador, Bahia, Brazil, 7-11 April 2008.
- [46] Guo, S., Keeney, J., O'Sullivan, D., Lewis, D., "Adaptive Semantic Interoperability Strategies for Knowledge Based Networking ", in *proc International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS '07)* at OTM 2007, Vilamoura, Portugal, 27-29 November 2007.
- [47] Keeney, J., Lewis, D., O'Sullivan, D., "Benchmarking Knowledge-based Context Delivery Systems", in *proc ICAS06*, Silicon Valley, USA, July 19-21, 2006.