Adaptive Semantic Interoperability Strategies for Knowledge Based Networking

Song Guo, John Keeney, Declan O'Sullivan, David Lewis

Knowledge & Data Engineering Group (**KDEG**)
Centre for Telecommunications Value Chain Research (**CTVR**)
School of Computer Science & Statistics, Trinity College, Dublin, Dublin, Ireland.
{gsong | John.Keeney | Declan.OSullivan | Dave.Lewis}@cs.tcd.ie

Abstract. A Knowledge Based Network is a type of ontological content based network. As Knowledge Based Networks scale, semantic interoperability becomes an important issue since larger populations of users result in more heterogeneity in the content of messages. This paper examines the content heterogeneity problem in a KBN and proposes a mapping service scheme for distributed and heterogeneous knowledge-based applications. It compares a number of strategies that use pre-existing semantic mapping information stored in KBN routers. Evaluation results show that this scheme can effectively solve the heterogeneity problem.

1 Introduction

Given the rapid evolution and dynamism of networking, there is increasingly a desire to allow applications which were designed independently and using different information structures to communicate that information without the necessity of custom building gateways. Publish-Subscribe (Pub / Sub) systems [1][2] provides decoupling of identify between producers and consumers of transmitted information, but requires messages to be categorised into predefined types. In response, Content-Based Network (CBN) have been developed [3][4][5]. These match messages to consuming client interests by specifying a filter on the messages' attribute values. The limitation of current CBNs is that they only support a very limited range of datatypes and operators for use in matching consumer subscriptions to message attributes, typically: Strings, Integers, Booleans, and associated equality, greater than, less than, and regular expression matches on strings. For a CBN to work on a large scale it needs to support a richer expressiveness that can cope with the widely heterogeneous and frequently changing range of message content and consumer subscriptions.

In previous papers [6][7][8] we have described a semantic-based CBN called the Knowledge Based Network (KBN). 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. An implementation of such a KBN, based on the Siena CBN [3], is available, and enables the efficient distributed routing of distributed heterogeneous knowledge to, and only to, nodes that have expressed a specific interest in that knowledge. We have been investigating the applicability of such Knowledge-Based Networking in the areas of Network & Telecoms Service Management, Autonomic Systems and Pervasive Services, Context & Management. In particular we have focused on semantic interoperability [9], self-managing networks [8][9], autonomic communications [10], context distribution [8][9], distributed service discovery [6][11], and the management of efficient knowledge routing mechanisms [12].

The majority of these investigations to date have assumed that the publisher and subscriber applications share a single common ontology. In this paper we go further and report upon how we have initially extended and tested the KBN so that it can cope with the situation where applications may be using multiple diverse ontologies. We have previously demonstrated [9] how the use of ontology and ontology mapping techniques enabled applications built according to different standards could interchange fault alarms over a CBN (Elvin [4]) using an ontology based approach. In that work ontology mappings were used in a generic gateway which was external to the network, co-located with the application. The work described in this paper is similar in intent but differs significantly in design and implementation, in that the ontology mappings are used directly in support of information routing within the network. Thus in this work the mappings are used to help route information within the KBN as well as supporting the model translation for applications at the edges of the network.

The incorporation of semantic interoperability strategies within the KBN routers means that applications that subscribe to information according to one ontology can expect to receive information published according to a different ontology, if there exists a mapping between the ontologies. Although this feature lowers the barrier for participation by applications in any particular KBN, it will potentially increase the workload of an individual KBN router. Our hypothesis reported here is that the impact of the extra processing is far outweighed by the benefits from enabling semantic interoperability between applications.

The enhanced KBN described in this paper supports the following requirements: (i) dynamic networks: where new applications can join in and leave dynamically and frequently; (ii) application autonomy: each application is responsible for its own information specification and representation; (iii) absence of a need for a-priori agreement between applications about information specification and representation.

Section 2 discusses semantic mapping and outlines the scenario used in the paper relating to the exchange of policies related to Dynamic Spectrum Access in a self-managing network. In section 3 the KBN router is described and several strategies for coping with mappings in the KBN router are identified. Section 4 presents some performance comparisons of the mapping strategies and discusses the factors that impact strategy selection and performance. Finally, conclusions and future work are outlined in section 5.

2 Semantic Mappings and Scenario

Semantic Mapping is defined as the establishment of correspondences between a set of source ontologies. In our work we assume that the ontologies are expressed in the web ontology language (OWL) [13]. OWL is also used to describe the mappings of the ontologies. In particular, we use *equivalence*, *subsumes* and *subsumed by* relationships to express the mappings[1]. In order to generate mappings, various matching techniques can be applied to the ontologies. However, the fully automatic generation of mappings from different ontology

[1] The *subsumes* relationship describes the super-class and super-property relationships. *Subsumed by* captures the sub-class and sub-property relationships. Equivalence can be used with classes, properties and individuals.

information is generally considered impractical [14]. This is because there is a degree of uncertainty in any automatic approach to matching two ontologies, with this uncertainty caused by the different syntactic representation of the ontologies, the combination of the similarity measures produced by different matchers, and the heuristic approaches inherent in some matchers [15]. For now, semi-automatic techniques for creating mappings from ontology matching information will continue to dominate [16] and in our work we have used the OISIN tool [17] to support the generation of the mappings.

As an example scenario, we have chosen the exchange of policies between service providers in a Dynamic Spectrum Access (DSA) environment. The XG policy language is developed by DARPA XG group for DSA management [18]. This is a rich scenario as the policy approach advocated by DARPA XG is ontology based, but it is unlikely that all service providers will develop policies according to one single ontology. Thus in order to support policy exchange between the service providers, an information delivery mechanism capable of supporting heterogeneous ontologies is required. Service providers may be arbitrarily located in different regional areas, so they need to subscribe their interests for policies relating to specific geographical locations.

In order to illustrate this approach, we give a concrete mapping example within our scenario, however, the approaches discussed in this paper are independent of the actual contents of the ontologies. Fig. 1 describes a hierarchical structure of mapping relations between classes and properties from two ontologies. The ontologies *xgpl-regn1* and *xgpl-regn2* are region description ontologies, made by the authors, for delivering XG policies between service providers (acting as KBN clients) in a KBN. To make a mapping between these two ontologies we first need to import both so that the rest of ontology description will be able to refer to the existing elements that are previously defined in an involved ontology. Second, we establish mappings between elements of the involved ontologies. For instance, one class of an ontology may be considered as a *subclass* of another class of another ontology (*xgpl-regn1:Village* is subclass of *xgpl-regn2:RuralUnit* in Fig. 1). Finally, two relations (*subsumption* and *equivalence*) between properties from the involved ontologies can be determined by comparing their members (*xgpl-regn1:ishouseof* is a subProperty of *xgpl-regn2:isbuildingof* in Fig. 1).

Now, let us assume that *xgpl-regn1* is the main application ontology distributed among some KBN routers. If there is a service provider interested in polices within a city range, it subscribes a query expressed by concept *xgpl-regn1:City* to its closest KBN router. If this router receives a

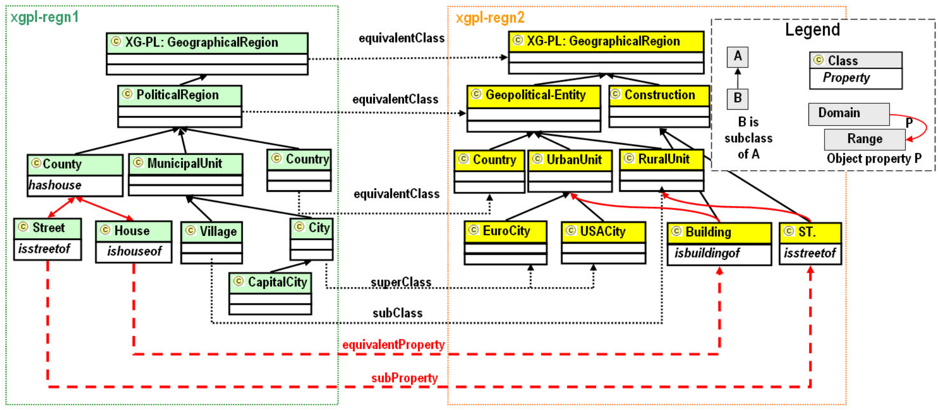


Fig. 1. Mappings between XG region ontologies

notification that has the same queried attribute name but the concept is *xgpl-regn2:EuroCity* (“*EuroCity*” is not defined in *xgpl-regn1* but rather in *xgpl-regn2*), this KBN router needs to explore the mappings to find mapping relations containing “*EuroCity*” and “*City*” to resolve this unknown concept. In this case, the region mapping ontology is explored, where the concept “*City*” is identified as superclass of the concept “*EuroCity*”.

3 KBN Router Model and Semantic Mapping Strategies

Before discussing how the KBN router model was extended to support semantic interoperability, the original KBN router model must be briefly discussed. The KBN router is an extension of the Siena content-based router [3]. A Siena notification is a set of typed attributes. Each attribute is comprised of a name, a type and a value. Siena supports the following attribute types: string, long, integer, double and boolean. A Siena subscription is a conjunction of filtering attribute constraints. A constraint is comprised of the attribute name, a comparison operator, and a value. A subscription covers a notification if the event satisfies all filtering constraints of a filter. A notification is delivered to a client if the client has submitted a subscription filter that covers that notification. Siena also discovers coverings between filters to optimise the subscription tree (subtree) at each router. As new subscriptions arrive at a router the subscription tree is searched to find the appropriate position to insert the new subscription.

By extending the Siena CBN model, the KBN supports 3 new ontological types: OWL classes (concepts); OWL properties; and OWL individuals (concept instances). The KBN also supports three new transitive operators for these types: “ontologically more specific” (MORESPEC); “ontologically less specific” (LESSSPEC); and “ontologically equivalent” (EQU). To achieve this, each KBN router holds a copy of an ontology, within which each ontological class, property and individual is described. A more detailed discussion of the KBN router model, and how it is extended from the Siena CBN router, is presented in [7].

3.1 Extended KBN Router Model

This section describes how the KBN router was further extended to support heterogeneous ontologies in the network. Each extended KBN router (fig. 2) is implemented with two ontology repositories: the main application ontology store provides the ontology for the KBN operation, whereas the mappings in the mapping ontology store are used for helping the KBN router achieve semantic interoperability. In the original KBN router, every router had a copy of the same main application ontology, however in this extension each router can have a different local main application ontology, and a different set of mapping ontologies to support interoperability between application ontologies. All ontologies are provided by the administrators of the network. The ontology registration interface allows administrators to register both application ontologies and mappings with KBN routers. Both publishers and subscribers register with KBN router via a client registration interface, and they need to provide their own ontology that defines the knowledge bases used by the clients in their subscriptions and notifications.

Subscriptions can arrive at the KBN router either directly from a client or from another node in the KBN network. The query subscription, using terms from the subscriber’s local ontology, is passed to the subscription tree (subtree) searching engine, which searches the subtree and inserts the subscription in the appropriate position. However, the subscription may use ontological terms that are not contained in the router's local ontology, and so the position to insert the subscription into the subtree cannot be immediately resolved.

Similarly, when a publication arrives at a KBN router, either directly from a client or from another KBN node, the subtree searching engine walks the subtree to find appropriate matching subscriptions to find the set of subscribers (clients and other KBN nodes) that should be notified with the publication. Again, the publication may use ontological terms that are not contained in the router's local ontology, so the set of matching subscriptions cannot be immediately resolved.

If the subtree searching engine receives a subscription or notification with ontological terms which are not expressed by terms from the application ontology, the mapping management interface is called to explore the mapping store where the mappings were previously injected. The next section discusses a number of different strategies that the mapping management tool can employ to handle unknown ontological concepts, properties or individuals.

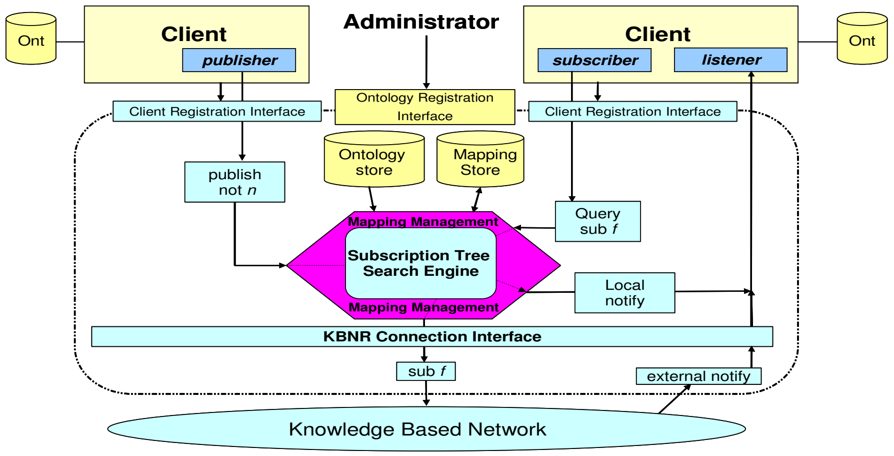


Fig. 2. Enhanced Knowledge Based Network (KBN) router architecture

3.2 Semantic Mapping Strategies

The multiple different application ontologies determine the diversity of mappings which are generated from these application ontologies, so different mappings may be stored in different KBN routers. Furthermore, based on the scale of the KBN, which can range from enterprise-scale to internet-scale, different routers may store different numbers of mappings. Thus the efficient utilisation of mappings in KBN has become a key challenge. For specific application domains of KBN, such as Network Service Management, there may be one or small numbers of mappings relevant to management information needed to store in the KBN router. Alternatively, for generic applications in a large-scale deployment, there may exist many mappings. Our previous work has indicated that the loading of new ontologies into the reasoner embedded in a KBN node is computationally expensive; the ontological reasoning is memory intensive; and memory usage is proportional to the number of concepts and properties loaded into reasoner [12]. Thus different strategies are required to cope with different situations in the

KBN, and it must be possible to alter mapping strategies to cater differing and changing scenarios.

Several strategies have been developed and implemented in KBN routers, aiming at efficiently exploiting mappings to achieve semantic interoperability. In Fig. 3, we outline the workflow of mapping strategies that are implemented in the KBN router. These strategies are focussed selecting available mapping ontologies (and ontologies referenced in the mappings), which can then be loaded to handle unknown concepts (or properties, or individuals). Strategy selection should be based on a set of rules, for instance, the rules might be defined according to characteristics of application ontologies or the rate of subscriptions and publications. This aspect of the work is not further discussed in this paper, but is a focus on ongoing work.

Strategy1: When a KBN router encounters an unknown concept (or property, or individual), it loads all known mapping files and their imported ontologies at once. This strategy maximises the exploration of mappings to tackle the heterogeneity problem; especially when there is a small number of mapping files stored in KBN router. This strategy would also reduce the probability that further unknown concepts will be encountered at a later stage, at the expense of a high once-off cost. However, this strategy may be wasteful if there are a lot of mappings in different files, or if the occurrences of unknown concepts are rare.

Strategy2: This strategy searches the mapping files stored in the KBN router in order to select mapping files which contain at least one concept used by the conflicting subscription or notification. The router then loads those selected mapping ontology files without necessarily loading, reasoning and merging all of the available mapping files. Furthermore, when compared with strategy 1, the KBN does not load the ontologies imported by the mapping ontologies. Because many of these imported ontologies may not be relevant and to load all imports could introduce considerable overheads. This strategy should be beneficial where KBN router stores a large number of specific and fine grained mapping ontologies.

Strategy3a: This strategy also searches the set of available mappings to try to determine which mappings to load into the KBN router's application ontology. Here only the individual

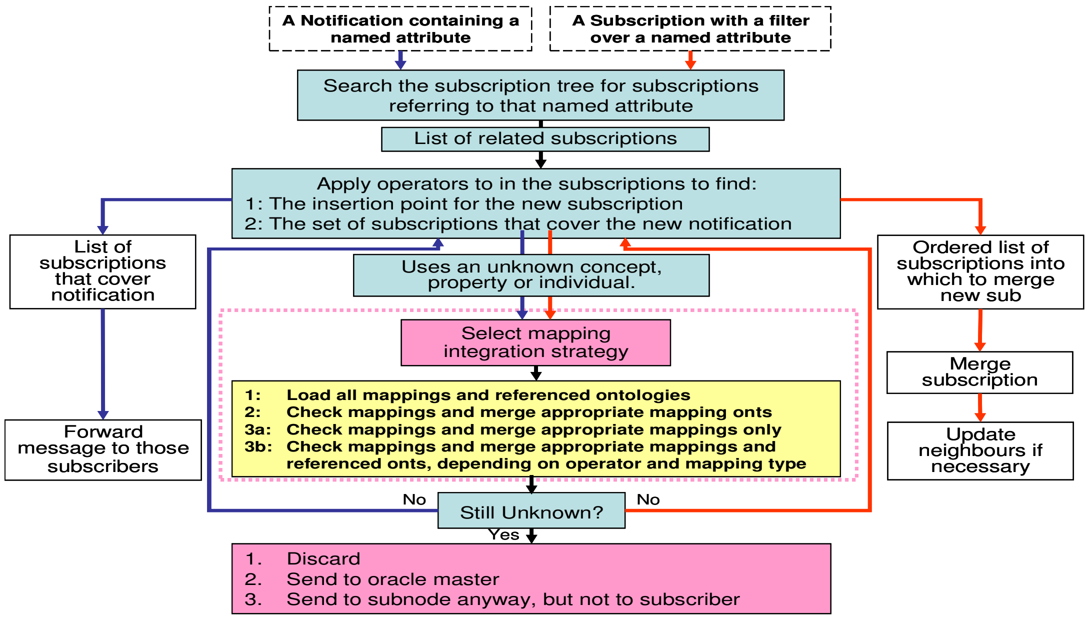
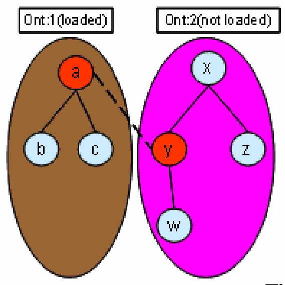
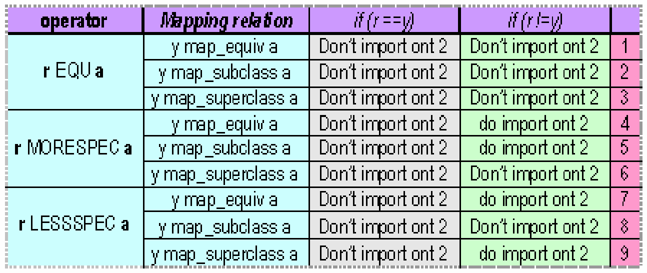

A Notification containing a named attribute
A Subscription with a filter over a named attribute
Search the subscription tree for subscriptions referring to that named attribute
List of related subscriptions
Apply operators to in the subscriptions to find:
1: The insertion point for the new subscription
2: The set of subscriptions that cover the new notification
List of subscriptions that cover notification
Forward message to those subscribers
Ordered list of subscriptions into which to merge new sub
Merge subscription
Update neighbours if necessary
Uses an unknown concept, property or individual.
Select mapping integration strategy
1: Load all mappings and referenced ontologies
2: Check mappings and merge appropriate mapping onts
3a: Check mappings and merge appropriate mappings only
3b: Check mappings and merge appropriate mappings and referenced onts, depending on operator and mapping type
No
Still Unknown?
No
Yes
1. Discard
2. Send to oracle master
3. Send to subnode anyway, but not to subscriber

Fig. 3. Workflow of Mapping Strategies

mapping is incorporated into the application ontology, rather than the whole mapping file it was contained in. Ontologies referred to or imported by the mapping ontology are again not loaded. This strategy only succeeds if a direct mapping exists between the unknown concept and a known concept. For example, based on the ontologies introduced in section 3 above, where the *xgpl-regn1* ontology is already loaded, but the *xgpl-regn2* ontology is not loaded. If the unknown concept *xgpl-regn2:EuroCity* from the *xgpl-regn2* ontology is encountered, the mappings are searched. A mapping is found linking the unknown *xgpl-regn2:EuroCity* concept to the previously known *xgpl-regn1:City* concept. In this scenario, only the mapping is loaded, but neither the *xgpl-regn2* ontology, nor referenced ontologies, are loaded.

Strategy3b: This strategy is very similar to strategy 3a above. However, unloaded ontologies referred to in the mapping may also be loaded, depending on the combination of mapping relation found, and the operator to be applied to the unknown concept (for subscription sorting or subscription matching). This strategy could be used where there may not exist a direct mapping relationship between the unknown concept and any known concept, yet the unknown concept may be present in the referenced ontology. Fig. 4 describes when the router needs to load the second referenced ontology. In Fig. 4, **r** is the unknown concept, and a mapping exists between **a**, a known concept in the loaded ontology, and **y**, a concept in an ontology referenced by the mapping. We have considered the equivalence, subclass and superclass relationships and the three new KBN operators, EQU, MORESPEC and LESSSPEC. Lines 4, 5, 7 and 9 shows the cases when the router needs to load second ontology to deal with the unknown concept **r**.





operator	***Mapping relation***	*if (r ==y)*	*if (r !=y)*	
r EQU **a**	y map_equiv a	Don't import ont 2	Don't import ont 2	1
	y map_subclass a	Don't import ont 2	Don't import ont 2	2
	y map_superclass a	Don't import ont 2	Don't import ont 2	3
r MORESPEC **a**	y map_equiv a	Don't import ont 2	do import ont 2	4
	y map_subclass a	Don't import ont 2	do import ont 2	5
	y map_superclass a	Don't import ont 2	Don't import ont 2	6
r LESSSPEC **a**	y map_equiv a	Don't import ont 2	do import ont 2	7
	y map_subclass a	Don't import ont 2	Don't import ont 2	8
	y map_superclass a	Don't import ont 2	do import ont 2	9

Fig. 4. Conditions for loading a referenced / imported ontology

For example, the KBN router gets a subscription containing a named attribute filter using the LESSSPEC operator over a known concept *xgpl-regn1:City* (**a**), and a notification arrives containing an attribute with the same name with its value an unknown concept (**r**) *xgpl-regn2:RuralUnit*. After checking the mapping files, the router finds that *xgpl-regn1:City* (**a**) has a subclass *xgpl-regn2:EuroCity* (**y**). At this stage, since the mapping has not yet been incorporated, it is unknown if *xgpl-regn2:RuralUnit* is related to *xgpl-regn2:EuroCity* (i.e. **r** = **w**, or **r** = **x**, or **r** = **z**). Thus the *xgpl-regn2* ontology (**ont2**) referenced in the mapping must be loaded. Afterwards, operation (*xgpl-regn2:RuralUnit* LESSSPEC *xgpl-regn1:City*) can complete, returning false, and the KBN router's operation can proceed as normal.

If none of the strategies above resolve the unknown concept, property or individual, the KBN is left with a number of possible alternatives for dealing with the unknown ontological data. These include: discard the message with a warning; forward the message to a predetermined "oracle" node that would be responsible for such unrecognised knowledge; or forward the message (publication, subscription, unsubscription etc) to its neighbouring nodes (but not directly to subscribers), hoping that they can handle it. This aspect of KBN routers' operation is not further discussed in this paper, but is a focus on ongoing work.

4 Performance Comparisons of Strategies

In order to determine which of the implemented strategies suit different applications scenarios, it was decided to undertake an experiment to compare performances of each of the strategies. Several metrics are measured in our experiments. Load-time overhead is given as the times taken for the reasoner to load, parse and consistency check the ontology, perform TBox classification, perform ABox realisation and an initial query of all concepts [12]. We also measured the time taken to check mapping files in the mapping store in order to find correct mappings ("check mapping list") and the time taken to merge these mappings. We also measured the size of the application ontology (number of statements in the ontology after it was reasoned over) after performing each strategy to measure the relative memory usage by different strategies[2].

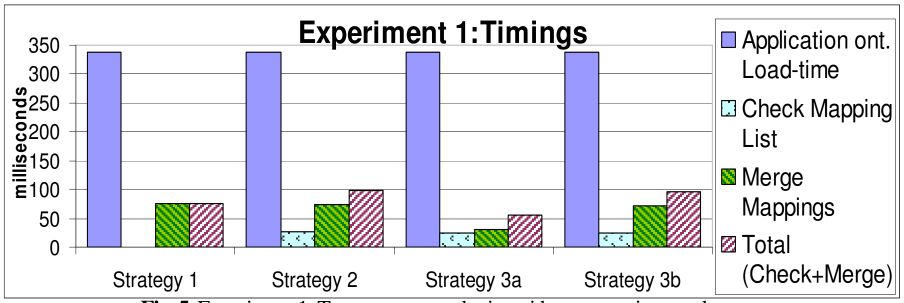


Fig. 5. Experiment 1: Two *xg-regn* ontologies with one mapping ontology

The first experiment compared the strategies by using the two *xg-regn* ontologies and the mapping ontology described in section 3. The *xg-regn1* ontology is stored in the application ontology store in a KBN router. The mapping store of the KBN router contains the region mapping ontology, whereas the *xg-regn2* ontology is used as a source of unknown concepts for the experiment.

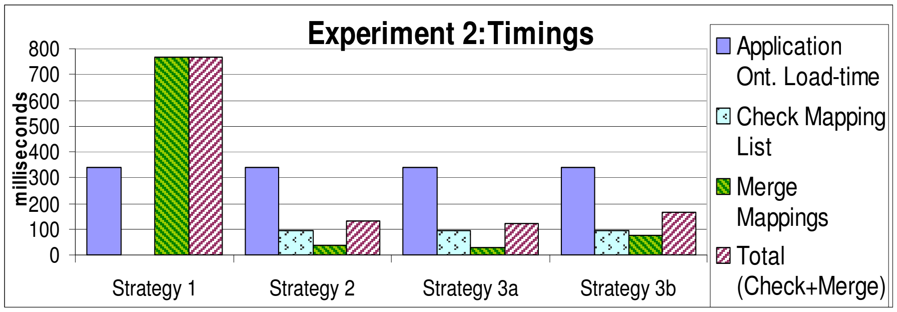


Fig. 6. Experiment 2: Three *xg-regn* ontologies with multiple mapping ontologies

In the second experiment, the *xg-regn1* ontology is again used as the application ontology and region mapping ontology is stored in mapping store, another three mapping ontologies

[2] All tests were taken on a Dell Inspiron 9300 laptop with 1.73 GHz Intel processor, 1GB of RAM, running Windows XP SP2. For java-based tools, Sun's JSDK 1.6.1 was used. Jena 2.3, with Pellet 1.3, was used for ontology manipulation. Tests were repeated at least 10 times. Reported timings are averages.

were chosen for testing. Firstly, we created another region mapping ontology, which was generated from the *xg-regn2* ontology and a third *xg-regn3* ontology. The second mapping data set is the owl-s mapping ontology [19]. It is used for policy management in semantic web services, and contains a large number of imported ontologies. Finally, the WOB ontology is relative small mapping ontology without imported ontologies [20].
As can be seen from Figs. 5 and 6, the time taken for a KBN router to load the main application ontology at network set-up time same for each strategy.

In Experiment 1, with only one mapping file, strategy 1 performed only a little worse than strategy 3a, since unlike strategy 3a the *xg-regn2* ontology that is imported by the mapping ontology was also loaded and merged. By using strategies 2, 3a and 3b the KBN router only sometimes needed to load the referenced ontology. Compared to other strategies, strategy 1 did not check the mappings to select which mappings to merge, since all available mappings and referenced ontologies are loaded, and so strategy 1 outperformed strategies 2 and 3a. Strategies 2, 3a and 3b performed similarly for the "check mapping list" time.

In Experiment 2 we see that as the number of mapping files in the KBN router increases, the performance of strategy 1 for mapping load time worsens dramatically. This is due to it having to load and merge more mapping ontologies and their imported ontologies. Strategy 3b performs a bit worse than strategy2 and strategy3a, because it merges referenced ontologies into the application ontology when the direct mapping relations between known and unknown data are found in mapping file. We can also conclude that the increased number of mappings does not have a major effect on the performance of strategies 2, 3a and 3b, just increasing the "check mapping list" time.

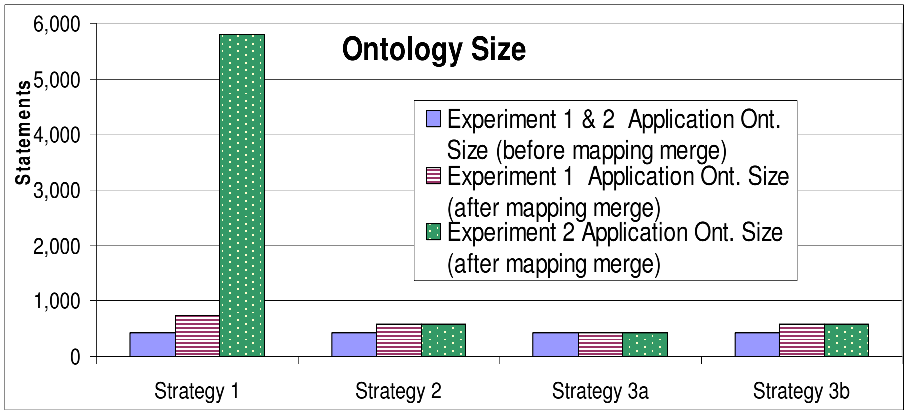


Fig. 7. Experiment 1 & 2: Ontology size

From Fig. 7 we can see the dramatic effect on resources when strategy 1 is used. For this reason it is obvious that strategy 1 should not be used where there is a large number of mapping ontologies, with a large number of imported ontologies, especially where the ontologies are particularly large or resources are scarce. With respect to size, strategies 2, 3a and 3b are much more efficient as only the correct mapping ontology is loaded. Strategy 3a performs best since the mapping ontology is not merged, but just the individual mapping relationship. Strategy 3b performs very similar to strategy 2, except depending on the mapping relationship and operator used an imported ontology may also need to be loaded.
Overall, across all of the experiments, strategy1 seems to have performed worst, with each of the other strategies performing similar to each other. However, it must be remembered that for

strategy 1, despite the once off cost of loading all of the mappings and referenced ontology, the KBN is much less likely to encounter unknown concepts in the future, thereby reducing the need to incorporate mappings at a later time. Similarly, although strategy 3b performs slightly worse than strategy 2 or strategy 3a, strategy 3b is more likely to load a complete referenced ontology. Overall, strategy 3a slightly out-performs strategy 2 and strategy 3b since it performs the least amount of ontology loading, querying and merging. This strategy is preferable where the occurrence of unknown concepts is rare and localised. Another point to remember is that the strategies that do not load the imported or referenced ontologies can result in the case where the unknown concept remains unknown after checking the mappings, and so the publication or subscription cannot be handled locally in a satisfactory manner.

5 Conclusions and Further Work

In this paper, a semantic mapping service scheme has been proposed to solve the heterogeneity problem in knowledge based networks by appropriately and efficiently selecting a mapping strategy efficiency of knowledge delivery in KBN. Several strategies are identified to explore how semantic mapping information can incorporated into the router's knowledge base. Each of the strategies were discussed and compared, with scenarios identified to suggest when different strategies should be selected. The comparative evaluation of each strategy indicates that different mapping strategies should be configured in different KBN routers depending on the particular characteristics of the routers' ontologies, the characteristics and number of available mappings, and the type of application operating over the KBN. In a small scale or enterprise scale scenario it may be possible to examine the application running over the KBN to statically determine which strategy is most appropriate. However, in a large scale deployment, or where the characteristics of applications using the KBN may change, then it is necessary to dynamically manage and adapt which strategy is most appropriate.

Work is ongoing to build on these initial evaluations 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 foreseen that the KBN itself would be ideal for such a distribution mechanism.

This work also builds on and validates work previous work by the authors [12] which discusses the impact of changing a KBN router's application ontology at runtime. From that work, and the findings presented in this paper, it is clear that where possible, the introduction of new knowledge into the KBN routers ontology should be minimised where possible, or performed at load time. However, this paper focuses on scenarios where this is not possible, either due to the dynamic nature of the applications, publishers and subscribers that use the KBN, or indeed where it is impractical or impossible (perhaps due to resource constraints) to have the entire knowledge base of the network statically replicated on every KBN router.

Ongoing work is also focussing on how the semantics of the knowledge in the network can be exploited to cluster nodes that focus on semantically similar knowledge [12]. In this way the interoperability of clusters' knowledge bases can be localised to edge of clusters, thereby localising the overhead required for semantic interoperability and mapping.

Acknowledgement. This work is funded by Science Foundation Ireland under Grant No. 03/CE3/I405: Centre for Telecommunications Value-Chain Research (CTVR) and Grant No 05/RFP/CMS014.

References

1. Meier, R., Cahill, V., “Taxonomy of Distributed Event-Based Programming Systems“, The Computer Journal, **48**(5), pp 602-626, 2005
2. Eugster, P. Th., Felber, P.A., Guerraoui, R. , and Kermarre, A.-M., “The many faces of publish/subscribe”. ACM Computing Surveys 35(2) (2003) 114-131
3. Carzaniga, A., Rosenblum, D. S., and Wolf, A. L., “The Design and Evaluation of a Wide-Area Event Notification Service”, ACM Transactions on Computer Systems, **19**(3), Aug. 2001.
4. Segall, B. et al, “Content-Based Routing in Elvin4”, In proc AUUG2K, Canberra 2000.
5. Pietzuch, P., Bacon, J., "Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware". DEBS'03 at ACM SIGMOD/PODS Conference. California, June 2003.
6. Lynch, D., Keeney, J., Lewis, D., O'Sullivan, D., "A Proactive approach to Semantically Oriented Service Discovery", Workshop on Innovations in Web Infrastructure (IWI 2006), at Int’l World-Wide Web Conference, Edinburgh, Scotland. May 2006.
7. Keeney, J., Lynch, D., Lewis, D., O’Sullivan, D., "On the Role of Ontological Semantics in Routing Contextual Knowledge in Highly Distributed Autonomic Systems", Tech. Report TCD-CS-2006-15, Dept of Computer Science, Trinity College Dublin. 2006
8. Keeney, J., Lewis, D., O'Sullivan, D., "Ontological Semantics for Distributing Contextual Knowledge in Highly Distributed Autonomic Systems", Journal of Network and System Management, **15**(1), March 2007.
9. Keeney, J., Lewis, D., O'Sullivan, D., Roelens, A., Boran, A., Richardson, R., "Runtime Semantic Interoperability for Gathering Ontology-based Network Context", Network Operations and Management Symp. (NOMS 2006), Vancouver, Canada. April 2006.
10. Lewis, D., O'Sullivan, D., Feeney, K., Keeney, J., Power, R., "Ontology-based Engineering for Self-Managing Communications", Workshop on Modelling Autonomic Communications Environments (MACE 2006), at Manweek 2006, Dublin, Ireland, Oct 2006.
11. Roblek, D., "Decentralized Discovery and Execution for Composite Semantic Web Services", M.Sc. Thesis, Trinity College Dublin. Dec. 2006, Tech. Report TCD-CS-2006-66.
12. Lewis, D., Keeney, J., O'Sullivan, D., Guo, S., "Towards a Managed Extensible Control Plane for Knowledge-Based Networking", International Workshop on Distributed Systems: Operations and Management, (DSOM 2006), at Manweek 2006, Dublin, Ireland, Oct 2006.
13. W3C (2003) Ontology Web Language, http://www.w3.org/2001/sw/, Visited Jun 2007.
14. Noy N. F. and Musen, M. A. “Algorithm and tool for automated ontology merging and alignment”, 17th National Conference on Artificial Intelligence (AAAI-2000), 2000.
15. Cross, V., “Uncertainty in the automation of ontology matching”. 4th Int’l Symp. on Uncertainty Modeling and Analysis, 2003. ISUMA 2003. 135- 140, Sept. 2003.
16. Uschold, M., Grüninger, M., “Ontologies and Semantics for Seamless Connectivity”. SIGMOD Record 33(4): 58-64. Dec 2004.
17. O'Sullivan, D, Wade, V., Lewis, D., “Understanding as We Roam”, IEEE Internet Computing, Volume 11, Number 2, 2007, pp26 - 33
18. DARPA XG Working Group “DARPA XG Policy Language Framework, RFC”, version 1.0, prepared by BBN Technologies, Cambridge MA, USA, Apr 16 2004.
19. Owl-s map ontology, http://ontology.ihmc.us/SemanticServices/KAoS-OWL-S-Mapping.owl
20. L. Ding, Web of Belief Ontology http://daml.umbc.edu/ontologies/webofbelief/0.8/wob.owl