Using Ontologies to Support Customisation and Maintain Interoperability in Distributed Information Systems with Application to the Domain Name System

Nickolas J. G. Falkner Paul D. Coddington Andrew L. Wendelborn
School of Computer Science, The University of Adelaide, Adelaide, South Australia 5005

E-mail: jnick@cs.adelaide.edu.au

Abstract

Global distributed systems must be standards-based to allow interoperability between all of their components. While this guarantees interoperability, it often causes local inflexibility and an inability to adapt to specialised local requirements. We show how local flexibility and global consistency can coexist by changing the way that we represent these systems. The proven technologies already in use in the Semantic Web, to support and interpret metadata annotation, provide a well-tested starting point. We can use OWL ontologies and RDF to describe distributed systems using a knowledge-based approach. This allows us to maintain separate local and global operational spaces which, in turn, gives us local flexibility and global consistency. The annotated and well-defined data is better structured, more easily maintained and less prone to errors since its purpose can be clearly determined prior to use. To illustrate the application of our approach in distributed systems, we present our implementation of an ontologically-based Domain Name System (DNS) server and client. We also present performance figures to demonstrate that the use of this approach does not add significant overhead to system performance.

1 Introduction

We have developed a general approach to modelling distributed systems ontologically using knowledge domains[4, 5]. This was initially developed by identifying the key structures storing information in large-scale distributed systems and representing these within an ontology. Further work led to the development of a more complex ontological model that captures the relationships between data, the representation of data and the operational semantics of the transformation of that data by the server.

In this paper we describe how we have taken an existing widely distributed service, the Domain Name System (DNS), and extended its existing functionality using knowledge domains. This allows for interoperation of standard and non-standard system components through semantic alignment and also facilitates automated semantic annotation. This also can assist in the formation and efficient use of semantically-based virtual organisations.

The DNS[9, 10] stores and provides mappings as a global and hierarchical system, employing distributed management, extensive use of caching and a strong standards-based model for making changes to the system. The most commonly used of these mappings is the mapping from IP address to computer name and vice versa. There are many others available but, whenever a new one is added, the defining standards must be altered and the new change implemented across all of the deployed servers and clients. The standards-based model is slow to provide changes and final implementation can also take considerable time.

We describe data in terms of its structure and its representation. Data in servers exists in stored mode, in the defining files or database, and in transit-mode, where it is modified or encapsulated for transmission to a client or server. Thus we need to know both how the data is stored and how it must be transformed and represented at each point in its life cycle. The key concept is that both of these can be mutable depending on who is requesting the information or what a local site wants to do in response to a request, having identified the request issuer.

We also use knowledge domains to describe behaviour as operational semantics without having to recode the underlying server. Our example describes those aspects that we wish to keep for all DNS participants (the global community) but also allows the description of those aspects that we wish to modify locally. Our model is characterised by the description of the logical and representational aspects of the DNS, as well as the operational semantics. This leads towards a flexible description of server behaviour. These operational semantics describe DNS operation - how is the data to be manipulated, under what circumstances and do we add or manipulate metadata?

Proceedings of the Second International
Conference on Semantics, Knowledge, and Grid (SKG'06)
0-7695-2673-X/06 $20.00 © 2006 IEEE

Why change or augment the DNS, when it is so widely and successfully deployed? The first reason is that it is very easy to misconfigure a DNS server and any mechanism that prevents or limits misconfiguration is valuable. A poorly designed or misconfigured query agent can cause a great deal of additional traffic through querying all possible responders simultaneously or repeating queries to servers that have already indicated that they do not have the answer. To counter this, we can strictly specify protocol and control standards to minimise non-conforming behaviour. The strict centralised control of the specification of the DNS can detect errors in implementation but any extensions beyond the standard would, correctly, be identified as errors against the central description and ignored.

The second reason to make changes to such a well-established system is that, while the provider of information may not have an interest in semantic annotation or provide different data delivery options, downstream handlers may wish to annotate records or receive information in different ways. Recasting the DNS in terms of knowledge domains allows reasoning-based service alterations without necessarily requiring recoding. We also gain the significant advantage that we can separate core activity from the extended feature set and allow reconfiguration without breaking the standards.

In this paper, we describe our approach for recasting a distributed system in terms of knowledge domains and implementing this using shareable ontologies. We illustrate our approach with our implementation of a core set of the DNS, using DNS resource records. The implemented system conforms to global standards and also provides a set of local extensions to allow additional mappings or new behaviours for a well-defined group of nodes. We provide performance measurements to show that we can provide the additional functionality without significant penalty.

2 Background

The DNS can provide name to IP address mappings (A records), IP address to name mappings (PTR records), canonical names (CNAME records) for IP addresses so that multiple names map to one IP address, facilities to support global e-mail (MX records), location of name servers (NS records) and, more recently, even the location of particular network services within a domain (SRV records) [7]. These resource records make up the heart of DNS and are all found within the zone files that provide the DNS data that name servers in turn provide in response to client requests.

DNS has three major components, described in RFC 1034 [9]. These are the domain name space itself, name servers, and resolvers. Resolvers query name servers to resolve client requests and are usually found at system level.

Within the domain name space, zone files are used to store the data associated with a particular domain. A zone file is composed of resource records. Resource records (RRs) store information about the owner, the type of resource, the time to live, the protocol family and the type-dependent format data. RR Type Numbers are used to identify the payload of a message sent to or from a DNS server. Rather than use the mnemonic 'A' or 'PTR' to identify the record, a pre-defined table of mappings from name to number is used and this number is used to identify the payload.

2.1 Key technologies

Semantic annotation is the addition of metadata that describes the meaning, purpose or function of data in a system, where the metadata is specifically associated with data that already exists in, or is being added to the system. A significant advantage of extending the DNS with semantic annotation is that a large number of applications across the Internet could use this metadata to store or derive information that would normally be stored out-of-band. This extension allows the integration of the DNS with other systems that can use or add annotation to provide additional services to their user community. Such annotation allows the integration of the DNS into the framework of the Semantic Web without using wrapper software.

The Semantic Web uses the Resource Description Framework (RDF) and RDF Schema to define its component resources [8, 1]. RDF is a triple-based representation that stores triples in the form (subject, predicate, object). The subject of one triple can be the object of another triple so quite large, and complex, relationships can be encoded. RDF Schema defines the relationships between RDF items, including domain and range of properties and subclasses.

The Web Ontology Language (OWL) [11] is built on the Resource Description Framework (RDF), which is in turn built on XML [14], and is a W3C standard for producing web ontologies. In essence, XML describes the data, RDF describes the relationships between data and OWL provides a framework for more advanced reasoning and the inference of additional properties or memberships over the data. OWL extends the functionality of RDF to describe cardinality, transitive properties and the explicit statement of class and individual equivalence. OWL has three distinct sub-languages, OWL-Lite, OWL-DL and OWL-Full, which provide varying levels of expressiveness and decidability, depending on user requirement. OWL ontologies can be referenced remotely and extended at the remote site very simply, providing that the URI of the original ontology is known. The formal semantics of OWL also provide a mechanism for deriving the literal consequences of information contained in the ontology - facts which are not contained within the ontology but which are entailed by the ontology.

The semantic web builds on the World Wide Web

Proceedings of the Second International
Conference on Semantics, Knowledge, and Grid (SKG'06)
0-7695-2673-X/06 $20.00 © 2006 IEEE

(WWW) by associating metadata with the data stored on the WWW to allow improved reuse and data sharing [6]. Metadata relationships, classes and properties can be captured in taxonomies or, with the formal specification of class relationships and detailed metadata characteristics, as ontologies. The semantic web allows the use of metadata to place a structure on the data stored and used within a system, and to show the relationships with the data's accompanying metadata. Ontologies can then be used to interpret and classify the metadata.

2.2 Motivation

The data stored in the DNS is not overly human-friendly nor, beyond commented annotations in the zone files, can it easily be annotated as a large amount of the metadata associated with the data is entirely implicit. This limits the potential re-use and integration of the stored mappings and DNS facilities by other services that could take advantage of an annotated data stream. Increasingly, metadata is required for the more efficient use of data.

Currently, we cannot add additional mappings or features without either requiring a global change to the system or risking losing our ability to interoperate with other system members. There is no mechanism that allows DNS behaviour to be altered on a site-by-site basis for individual hosts while remaining strictly compliant with the defining standards required by all other sites and hosts. In this context, a site is a logical entity: it could contain one machine, several machines in one location or the machines of an entire company, spread nationally or globally. It is a close approximation to the notion of a virtual organisation (VO) referred to in Grid computing. In this paper, we show how to describe such behaviour at whatever size or nature of site is required. Existing work applying ontologies to the DNS [2] does not address the management of the information contained within the DNS.

3 Ontological representation of the DNS

We now briefly discuss our approach, starting with a description of the record handling requirements of the DNS. RFC 1035[10] defines the RDATA component of resource records, the component of the record that is returned in response to a query. All RRs have the same top level format with varying RDATA elements. Resource records are transmitted in the DNS message format [10].

We consider such records to have a logical relationship between themselves and the parent notion of a resource record. They also have a structural representation that defines how the value stored in the mapping is to be sent to a resolver or a server. In the case of an A record, this is a 32-bit internet address. In the case of a TXT record, this is a set of one or more <character-string>s, where a <character-string> is defined as a contiguous set of characters without interior spaces, or as a string enclosed in double-quotes. The behaviour of the server when a request is received is defined by its *operational semantics* - how it handles the incoming byte stream and produces a locally coherent request that it can then answer, recode and then send back to the requester.

We analysed all of the data structures and available record types, to produce an ontology that could by used as input to a modified DNS server[5]. This provides contextual information on the type and nature of data and simplifies the annotation process. Ontologically stored data is also significantly more human readable because well-named OWL and RDF tags clearly describe the roles of classes and properties. Traditionally, annotation in DNS is carried out using comments embedded in zone files. These are human readable but are not made available by the server. With annotation, human-friendly comments are parsed in and made available by the server in association with the standard data streams.

This illustrates the capability of the approach to provide more configurable access to the data stored within a system. Rather than being able to only generate a single logical stream of data, ontological configuration can allow a number of different data streams, with or without annotation, to provide the information to the client in the most useful form. Because the ontology can be used to describe the communication protocols, as well as the data structures and relationships, we can also make the data available in a variety of formats. In the DNS, for example, we could make the data available in DNS message format, SOAP or XHTML - depending on the client's requirements.

4 Semantic representation of the DNS

It was quickly apparent that producing an ontology that dealt only with data was restricting the potential benefits of this approach. Our first modified DNS server was, other than the format of its zone files, a standard DNS server. We began to look at the other possibilities of ontologies, in particular using an ontology to describe the behaviour of the server. In this section we discuss the representation of *server behaviour*, rather than just the data, and justify our language choices for the ontology and the representation of server behaviour.

We can use an ontology to represent correct server behaviour. To do so, we classify behaviours into classes and denote properties to show transitions between different behavioural states. We can marry this with the description of data and data relationships that is already in the system to produce a large, three-branch ontology. The three branches are: the logical relationships between entities, the represen-

tational aspects and the operational semantics that describe the transition from one state to another.

To represent the operational semantics of the DNS, we chose to use a modified form of the λ-calculus as it is well understood and provides excellent support for the definition and graph representation of anonymous functions. These graph structures can be embedded into RDF graphs and allow us to associate anonymous function structures with named functions in the operational semantics graph. Effectively, the λ-calculus expressions define transformation functions that rewrite the DNS data to produce a new byte stream. While we could use XLST, XQuery or XPath to carry out such transformations, using the λ-calculus allows us to form a graph that we can insert into the ontological representation of operational semantics and maintain an RDF graph form. This, in turn, allows us simpler mechanisms for function application and allows us to use the simple, yet powerful, reduction rules of the λ-calculus to simplify expressions while maintaining provably consistent representations of the same function.

We provide a set of built-in functions, supplied by the server implementation, that extract class members from an ontology. These determine if a tested element is a member of the specified class and this allows us to provide a class 'Internal' and add members to it, then test in the DNS server whether a request had come from a client that had an IP address that was a member of the Internal class. We also provided built-in functions that retrieve the requester's IP address to use for comparison. Thus, a virtual organisation (VO) can set up a virtual domain that spans all of its members and allows these members to access data and annotation information that is inaccessible outside the VO.

5 Advantages of representing operational semantics in an ontology

A standards-based system is, beyond conformance to updates in the standard, inflexible in that it may not move outside of the standards. The underlying software is often produced based on the assumptions implicit in the standard. For example, DNS servers use hard-coded references to resource record types since these cannot be added in an ad-hoc fashion. Adding a new RR requires the insertion of code and checking that this new code doesn't cause problems in the rest of the server. Moving the operational semantics of data transformation out to an ontology requires only that the server be able to interpret the ontology and use these transformations, in addition to any hard-coded segment of the code. Further changes can be effected by changing the OWL ontology and reloading it.

The relationships between data can also be exploited to automatically extract information that is not explicitly stated. For example, an A record mapping a name to an IP address could have an inverse relationship, using existing OWL language features, so that the IP to name mapping (PTR record) is generated automatically.

The DNS is strongly controlled (through the RFC mechanism) and this strong control ensures interoperability between different servers and clients. However, under the standardised DNS modification scheme, this control also means that an RFC process has to be followed to bring about a global change; this is a slow process. With a knowledge domain encoding of the DNS system, DNS modification is no longer a centralised, rigidly controlled and slow process. At the same time, it still provides core functionality based upon the RFCs. We support this by explicitly separating the core and extension ontologies to prevent the accidental removal of core features.

6 Implementation of an ontologically-based DNS server

We adapted an existing DNS server to parse XML-based data files, interpret the ontology and use the existing internal data structures to store the parsed data. We also implemented explicit user functions, where we could use as much or as little of the existing DNS implementation to allow the testing of user functions.

We implemented simple guard statements, conditional containers and anonymous functions to support simple operation while retaining the cryptographic mechanisms of the original server for existing authentication mechanisms[12, 3], as these mechanisms are an important part of the secure mechanisms used to protect and authenticate DNS data and prevent subversion.

We chose to base our system on the 'dnsjava' Java-based DNS implementation, written by Brian Wellington [13]. This system is a straight-forward implementation of the DNS RFCs in Java. dnsjava is not a commercial-grade DNS server but, because of the programming approach taken, it was easy to modify while still maintaining the required DNS functionality. Although the Berkeley Internet Name Daemon (BIND) was originally considered as a source base, the method it uses for encoding resource record types depended upon compile time pre-processor statements and would have required major modification.

The key to our modifications is that dnsjava uses an abstract Record class that all of the RR types extend. In order to remove the dependency on compile-time definition of RR types, we alter the server to use a GenericRecord class that extends Record. After instantiation, such a record is then modified based on information stored in the representational ontology to store the correct components of the correct type. Methods are provided to transfer data to and from wire format, and to and from String format, as per the original Record.

We also modified the storage of RR type numbers so that these were read from the ontology, rather than being predefined in a static structure, as is the usual approach.

In our implementation we first produced the OWL files describing the knowledge domains for the DNS and the instances of the classes giving the zone data for a test domain. We then used an existing DNS server and implemented an RDF/XML parser to read in the OWL files. We used this to demonstrate correct operation of the DNS using ontological configuration files. We then implemented the operational semantics knowledge domain as an OWL file and developed the simple command parser and λ-calculus parser to read in the functions. We used this to demonstrate new and explicitly defined functional behaviour, including adding new RR Types for a limited domain, and tested that they worked correctly. We developed a set of ontologically-enhanced tools that use the ontologies to increase the efficiency of data requests, updates and transmission.

We modified the server to check for the existence of functional definitions at start-up. Where the resulting expression could be easily parsed it was pre-reduced, for efficiency, and the result stored. Thus, if a record would always return data, regardless of who asked for it, then this was recorded and no further parsing took place in response to additional queries. Similarly, for a record that never returns data, we define an associated shortcut that ensured that the server never parsed it or sent data. Where an expression could have variable outcomes, these were always parsed based on the nature, identity and location of the querying agent - as far as could be determined.

7 Results

The implemented system was designed to show that our approach can maintain the defined DNS functionality while allowing ease of extension in a way that does not break the centrally defined authoritative model.

In order to test our approach, we developed an ontologically based version of the domain information groper (dig) tool. dig is used to provide a mechanism for querying and testing DNS entries. Our dig variant derives its knowledge of the DNS from ontologies, and can access shared ontologies to determine the DNS extensions in use by an enhanced server. We collected performance information by using the dig tool provided with OS X 10.3.9, also the ontologically enhanced dig tools, and performing queries against an enhanced server and a standard server. dig performs DNS lookups and displays the answers that are returned. We changed the operational semantics associated with certain requests to demonstrate the impact of local-site customisation for clients that were aware of the change and for clients that were unaware of the change. We refer to the transformation between a recognised DNS request and the returned response as the *functional mapping*.

Experiment	Average Time (s)	Std Deviation
1a	10.04	0.067
1b	0.085	0.017
2a	0.922	0.027
2b	1.124	0.046
3	1.092	0.024
4 (**baseline**)	0.089	0.007

Table 1. Performance measurements

Our test environment consisted of a mix of standard and enhanced servers and clients. We set out to compare the performance of the interoperation of the two types of software, and demonstrate their successful interoperation. There were four basic experiments carried out, with experiments 1 and 2 separated into two further divisions. Functional mappings could not be changed on a standard server, which is why experiments 3 and 4 do not have the a and b variants. Each experiment was conducted 1000 times and the mean and standard deviation were calculated. Table 1 shows the collected performance measurements.

In experiment 1a, a standard dig tool was used with an enhanced server and an A record was requested for which records existed. The functional λ-calculus mapping associated with the A record was set to the null mapping, ensuring that any internal response was mapped to a null response. This response was inserted into the DNS message format and delivered to the querying agent. This demonstrates what occurs when a bad response, in this case the unmodified null response, is sent back from a server. The authoritative server had been instructed to return a malformed response and, after 10 seconds, the query times out, as no servers have given a response. Experiment 1b shows the successful interaction of modified and unmodified systems using a standard dig tool with an enhanced server and requesting an A record for which records existed. The functional mapping was set to identity.

Experiments 2a and 2b show the function of the completely enhanced system, with both client and server having full access to the functional specifications of the stored records. In 2a, the client doesn't send a request because the dig tool can read the server's ontology, determine that the server will not send an answer and, hence, not send a query that will receive no answer. In 2b, it can send a request and does receive an answer, as the functional mapping reverts to the identity function. It takes approximately 0.6 seconds for the ontologically-enhanced, Java-based, dig tool to start, read in the ontology and then start processing the user request and, taking this into account, it is clear that not sending the request improves the efficiency of the client/server communication by approximately 100%. In both experi-

Proceedings of the Second International
Conference on Semantics, Knowledge, and Grid (SKG'06)
0-7695-2673-X/06 $20.00 © 2006 IEEE

ments, an ontologically enhanced dig tool was used with an enhanced server. In 2a, the functional mapping was set to null and in 2b the functional mapping was set to identity.

Experiment 3 shows the overheads of using the enhanced dig tool with a standard server. An A record was requested for which records existed.

Finally, experiment 4 is a traditional DNS client communicating with a traditional DNS server. A standard dig tool was used with a standard server and an A record was requested for which records existed. Performance is comparable with the performance of a standard tool with an enhanced server, as in experiment 1b.

From this, we can see that the choice of client and the implementation of that client may have some effect upon the overall time taken to resolve a query but problems introduced by misconfigured servers are far more significant.

Correcting such a misconfiguration in a standard server would take considerable time. Using the enhanced server, we can resolve a problem causing a 1000-fold delay in a matter of seconds without significant performance loss.

8 Future Work

We plan to implement all of the DNS RR types as purely ontologically specified entities rather than entities with explicitly defined server support. This includes the crypographically-based RR types that are crucial for secure operation, which at present undergo no semantic modification.

There is also an issue in advertising the new RR types where access to the defining ontology is not available. We are developing a DNS SRV based system that uses advertisements combining ontological statement of the operation with the location and functional definition of the new RR.

9 Conclusions

Starting from an ontology that represented only the data in a system, we have shown that an existing knowledge domain-based representation can be extended to capture the behaviour of a complex system, such as the DNS, without loss of functionality or significant performance loss. Additionally, the underlying data of the system can be more easily annotated, interpreted and placed into context because of the rich structure now associated with every item.

Our implementation also shows that the correct representation of knowledge domains can be used to define behaviour in systems and replace existing system configuration and control mechanisms. Not only is this without loss of existing function but it provides additional capacity for expansion and is a potential springboard for the next generation of annotated, context-aware distributed systems.

We have also discussed why a global system such as the DNS, which has a great burden of existing implementations, can make effective use of a local modification strategy that allows users far more freedom than under the standards mechanism. However, we have also shown a mechanism whereby both global and local schemes can happily coexist and potentially benefit from extensions in either area.

We believe that mechanisms of this type can be used to integrate older internet technologies with newer distributed computing initiatives, including the Semantic Web, Grid and Semantic Grid, without having to expend a vast amount of effort in developing parallel information mechanisms.

References

[1] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, 2004. http://www.w3.org/TR/rdf-schema.

[2] C.-S. Chen, S.-S. Tseng, C.-L. Liu, and C.-H. Ou. Building a DNS Ontology using METHONTOLOGY and Protege-2000. In *Proceedings of 2002 International Computer Symposium: Workshop on Artificial Intelligence*, volume 2, pages 1853–1860, Taiwan, 2002.

[3] D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535, 1999.

[4] N. J. G. Falkner. Towards a Semantically Enhanced Internet: Developing an Ontology for the Domain Name System. Technical Report DHPC-161, The University of Adelaide, October 2005.

[5] N. J. G. Falkner, P. D. Coddington, and A. L. Wendelborn. Developing an Ontology for the Domain Name System. In *Proc. of the 4th Intl. Workshop on Web Semantics*, Copenhagen, 2005. IEEE.

[6] D. Fensel, W. Wahlster, H. Lieberman, and J. Hendler. Introduction. In Fensel, editor, *Spinning the Semantic Web*. MIT Press, 2003.

[7] A. Gulbrandsen, P. Vixie, and L. Esibov. A DNS RR for Specifying the Location of Services (DNS SRV). RFC 2782, 2000. http://www.dns.net/dnsrd/rfc.

[8] F. Manola and E. Miller. The RDF Primer, 2004. http://www.w3.org/TR/rdf-primer.

[9] P. Mockapetris. Domain Names–Concepts and Facilities. RFC 1034, 1987. http://www.dns.net/dnsrd/rfc.

[10] P. Mockapetris. Domain Names–Implementation and Specification. RFC 1035, 1987. http://www.dns.net/dnsrd/rfc.

[11] M. K. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language Guide, 2004. http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[12] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845, 2000.

[13] B. Wellington. dnsjava 1.6.6, 2004. http://www.dnsjava.org/.

[14] F. Yergeau and et al. eXstensible Markup Language (XML) 1.0 (Third Edition), 2004. http://www.w3.org/TR/2004/REC-xml-20040204/.