4. Discovering Homecare Services

Liam DOCHERTY and Evan MAGILL
Computing Science and Mathematics, University of Stirling, Stirling, FK9 4LA, UK

Abstract. Future homecare networks will consist of a very wide range of embedded services and software that will often rely on numerous other components to achieve their tasks. They will rarely operate in a self sufficient manner. The ability to discover and use services is not however a trivial task. Services may provide raw data, such as temperature readings, or higher contextual data, such as user activity and availability. Networks may change over time and may not be subject to a single management regime, implying the need for a great deal of self-reliance for any software component seeking services from elsewhere within the network. This chapter describes work carried out at the University of Stirling to improve service discovery and allow it to operate effectively in networks with a significant turnover in services. Simple syntactical keyword lookups are insufficient, and so semantics are introduced into the discovery process by using ontologies. However ontologies are known to grow and change over time and so maintaining them can be difficult and error-prone. The described approach employs a hierarchical approach that fosters re-use and sharing of ontologies to alleviate some of the more acute problems of building and maintaining large ontologies.

Keywords. Service discovery, Ontology, Homecare networks,

Introduction

Affordable network consumer devices have resulted in many of us having small computer networks in our homes. Such home networks include simple entertainment and communication services [1, 2], lifestyle management [3], and the provision of care in the home [4]. Those with an interest in computing and networks may well configure these and set them up to link computers, laptops, game consoles, and printers with the internet. Others of us may struggle. In reality we are expected to become system administrators, albeit for a small network. With the use of Internet television this can only become more challenging. How can this linking of home devices be made easier and become a more reliable activity? Indeed how can this become automatic?

The automation of connecting devices together has been the subject of a large volume of research. The goal is to allow a new device to be placed in a home and for it simply "to work". Terms such as "plug and play" are often used to describe this ideal and indeed some progress has been made. However it involves a great deal of agreement and often does not work reliability with products from different manufacturers.

Such automation requires that different network components are able to work together. At one level this is achieved through common communication protocols such as those found in home wireless networks. However while this allows devices to talk to each other it does not promote the way devices can work together effectively. In reality a home network may have devices such as sensors, printers and computers, and in

some cases these devices may be further divided into smaller components. For example, a computer system may contain logically distinct components, such as a visual display, a webcam and a keyboard. Home networks may also have software running that wants to make use of these devices. Examples here might include measuring temperature, or looking for movement in a room. Many of us are familiar with a laptop running word processing software, and carrying out the task of trying to configure the software to use a suitable printer.

The software needs the ability to discover and utilize network components, which may be hardware devices but could also be other software. This discovery and interaction is typically invisible to the user, being automated as much as is feasibly possible. Often the components being utilized are described as *service providers*, where that service might be to print paper, or perhaps provide a room temperature. The interaction between a service provider and the user or client of the service can operate in a loosely coupled manner, where the clients need not be bound to a specific provider, and instead may use any service provider that is available.

The process that allows a client component to find a service provider component is called *component discovery*. This can take the form of advertisements where the provider tells all the client components what services it offers, or the client can use a local network component registry which acts rather like the familiar "yellow pages" directory.

In order not to limit the user or the network to a specific vendor or indeed protocol, software is often used to bind heterogeneous components together. This software is often referred to as *middleware*. Middleware frameworks are typically used to manage and support the Home Network infrastructure, providing discovery mechanisms for service users and component description registries for providers. Middleware also supports the interoperation of services, independently of the particular protocols used. In this manner, multiple protocols may exist in the same network where a client and a service provider may use different communication protocols. For each protocol, there is often a distinct international standard where each protocol will have a different set of descriptions, vocabularies, attributes and schemas. Schemas and vocabularies provide a language, or a terminology, for describing items. Attributes are used to assign descriptive properties from the language to a particular item. It is common to refer to such an item here as an "entity". It is simply something that needs to be described.

While description schemas and attributes can differ between protocol domains, the meaning of the attribute values are typically identical. In other words, the attributes may be written differently in that they have a different *syntax*, but actually have the same meaning or *semantics*. This issue is especially problematic within the Home Network environment, which is assumed to be an ad hoc network of interoperating devices and services [1]. A common approach to this issue is that the middleware framework should insist on the component providing a description based upon the middleware's vocabulary and that all components must adhere to this middleware standard. This chapter advocates a much more flexible approach.

For a Home Network to be truly ad hoc, the middleware framework should not presume a middleware-compliant component description exists. Instead, middleware frameworks should rely on there being a component description which is specific to the protocol or platform of the component. If this were not the case, the vision of "anytime, anywhere" service deployment and interoperability is limited.

This problem can be addressed by translating the component description on behalf of the component. In this manner, the framework accepts the component specific

description, and translates it for itself. However this approach creates an important issue. Translating from the component description to the framework schema requires the framework (or perhaps agents acting on behalf of the framework) to either *understand* the component description or have an understanding of the schemas used. For this approach to succeed, the middleware requires knowledge of the relationships between the terms used within the component description vocabulary and the terms used within the framework description vocabulary. It can then classify descriptions from external vocabularies into descriptions using its own vocabulary.

In computer networks it is possible to create an *ontology* which explains a set of concepts for a particular scope or setting such as a particular aspect of a home network. The ontology also explains the *relationships* between these concepts. Originally defined for the Semantic Web environment, ontology languages are increasingly found within the distributed computing environment. The Web Ontology Language[1] (OWL) is a Description Logic language which has been developed specifically for describing and uniting domains. Using such a language provides support for the classification of component descriptions across a number of domains.

This chapter describes an ontology-based approach using a description framework which provides universal classification in the Home Network and allows component discovery to operate independently from any particular protocol or middleware framework. This approach also allows the unification of protocols and middleware at the description level. This chapter describes a middeware implementation which applies this approach to a Home Network environment. Providing translation at the middleware side enables the classification of components from differing protocols and middleware frameworks. This provides an essential element to allow networks to support the automatic configuration of their components.

1. Ontologies

As introduced in the previous section, an ontology is able to describe concepts and relationships by employing an ontology language. Ontology languages are a developing area of logical languages. Logical languages have a rigorous mathematical formality designed to minimize ambiguity. They permit automatic checking of any system being described in the language. Ontology languages inherit these capabilities and indeed are designed explicitly to assist machines in decision making, through techniques such as classification, inference and reasoning. Consider each of the three techniques in turn:

- **Classification** means to assign an identity to a concept or entity based upon an existing identity. For example, if an entity is classified as a Car, it can be further classified as a Vehicle.
- **Inference** means to assign descriptive properties to a concept based upon an existing identity. For example, if an entity is a Car, it can be inferred that it must also have Windows, Wheels and Steering Control, given that these components would be common to all vehicles.

[1] The Web Ontology Language specification can be found at http://www.w3.org/TR/owl-features/

- **Reasoning** means to assign further classification based upon descriptive or contextual information. For example, suppose all that was known about an entity was that it had Windows, Wheels and Steering Control. It is possible to reason that the entity is a Car entity.

These terms are related in practice as classification may lead to inference, which in turn may lead to reasoning and further classification. Using ontologies, users can describe a domain in considerable detail, including the important concepts within that domain and the relationships with each other. For example, Figure 1 contains a simplified ontology description of a domain concerned with document classification. Computer programs in the form of "search agents" can then be programmed with this knowledge and deployed within a particular scope or setting. It is common to refer to such a setting as a *domain*. For example, if the domain was concerned with scholarly documents, an academic webpage could be defined as a type of document. This is shown in Figure 1, where the class "Webpage" is defined as a sub-class of document. If a software agent was searching the network for documents about service discovery, it may include a standard set of articles in its search, such as journal publications and conference proceedings. With knowledge about academic webpages embedded within it, the agent could then increase its search domains to include these webpages, as it knows they are also considered a relevant type of document.

```
<Class ID="Document"/>
<Class ID="ConferenceProceeding">
      <subClassOf resource="#Document"/>
</Class>
<Class ID="Journal">
      <subClassOf resource="#Document"/>
</Class>
<Class ID="Webpage">
      <subClassOf resource="#Document"/>
</Class>
```

Figure 1. Example Document Ontology.

2. Unifying existing frameworks

This section describes how ontology languages can provide a means of unifying existing network protocol and component descriptions. This approach is a novel way of combining a broad range of different protocols and components that originate from a wide range of sources. So rather than create a new stand alone description framework that will fail to encompass a sufficient number of descriptions, the aim is to build upon existing descriptions. Not only does this broaden the scope, but it also allows for updates and changes to be accommodated with greater ease. In other words the approach is to encourage a unification of existing frameworks, should they be ontology

based or not. This section will discuss the application of the approach to common home network existing discovery vocabularies, such as Universal Plug and Play[2] (UPnP). Section 6 discusses applications of ontology description frameworks outside of the home network.

Commonly, concepts contained within one existing description vocabulary are semantically similar to those in others; that is, they mean the same thing. This can be taken one step further, where ontology concepts and properties can be related to attributes and elements from existing protocol and middleware descriptive frameworks. For example, the imaginary concept *Google:Webpage* can be said to be identical to the concept *Yahoo:Webpage*. This way, descriptive attributes and concepts from a number of description frameworks can be classified together under a single classification concept. In order to unify these descriptive frameworks, one of the frameworks must be employed as a default or root description framework.

As there is a lack of complete open-source ontology frameworks, it was necessary to develop an exemplar set of ontologies that describe a range of devices and services commonly used within a Home Network. Rather than having a single ontology, a set of ontologies was formed describing a wide variety of components. These ontologies also include related meta-data about the main entities within the domain. Meta data can be thought of as further detail about a particular entity, such as the type of video output provided by a webcam. Crucially, this set of ontologies also captures elements from other middleware and protocol description frameworks. In general, there is a one to one mapping between terms within an existing description framework, and terms within the ontologies. In Figure 2 the ontologies are set within a Home Network Ontology Stack.

Developing such a stack hierarchy, rather than having a flat set of ontologies, is important as it allows ontology reuse. This enables concepts and terms developed at the lower levels to be reused and extended within the higher levels. For example, the Base contains a 'Device' concept, which is developed further within the Core level ontologies. [5] describes recent work with ontology reuse as a means to support decision support software, relying on ontology reuse to describe contextual environmental data from sensor input. As will be discussed later, ontology reuse provides increased compatibility with other existing ontology frameworks, as well as aiding in the development process.

In Figure 2 the Base Ontology level is a single ontology that simply states the basic concepts relevant to the Home Network domain; so concepts such as *Component*, *Device*, *Service* and *Location* are given. Building upon this foundation, the Core Ontology level contains multiple ontologies, each corresponding to a concept stated in the Base Ontology. For example, the Device ontology deals with concepts and properties relevant to (home) Devices. This provides the ability to express abstract statements such as *Device X, offers service, Service Y* and *Device X, has location, Location Y*. For example, using this knowledge set a *Television* device could be said to offer a *Video Display* service, and can be found in the *Living Room*. Classifications or enumerations of the Service concept are defined in the Service ontology, and similarly for Locations, within the Location ontology.

[2] The Universal Plug and Play specification can be found at http://www.upnp.org/

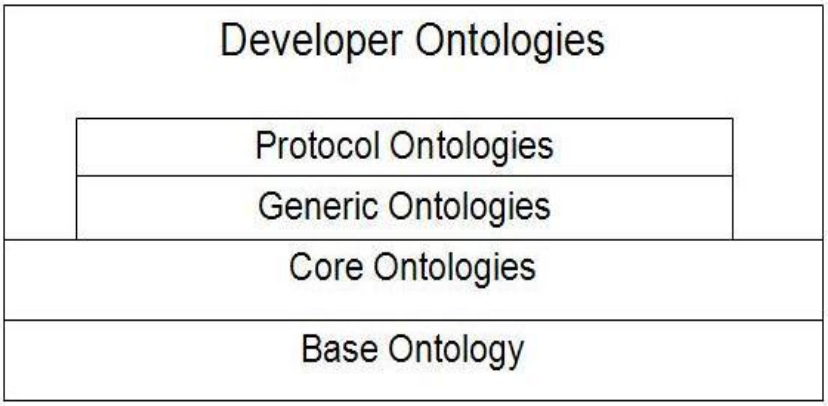

Figure 2. The Home Network Ontology Stack.

On top of the Core Ontology layer lies the Generic Ontologies layer which represent typical services and devices found within a home network. A generic device ontology is typically built upon a Device ontology and a Service ontology, found within the Core ontology level of the stack. Indeed it will reuse terms from the Service ontology to specify the kind of services a device can offer. In the generic layer, a minimal number of assumptions are made in order not to restrict the potential application of the ontologies. For example, a Lamp device offers a Lighting service, and is a sub-type of the more generic Light device. How a user (i.e. client software) would interact with a Lighting service is assumed to be protocol specific, and therefore beyond what is covered in this chapter. The Generic Ontologies provide the classification framework, which can be applied to external description frameworks. For example, if the framework is given a description of a device which offers a lighting service, by using the knowledge given in the Generic ontologies, the framework can infer that the device is (at least) a Light device.

The Protocol layer is concerned with relating protocol specific component properties to those contained within the lower ontology layers. For example, a Lamp device controlled by the UPnP protocol would be described as subclass of the Lamp component in the above example. Most descriptive properties within the UPnP specification, such as *manufacturer* or *friendlyName*, are related to properties contained within the core layers of the stack. Other UPnP specific properties, such as an event or a subscription URLs, are unique to the UPnP ontology. While having unique properties in higher levels would initially seem to negate the purpose of a universal description framework, it allows protocol specific information to be retained within the description. In this way, after discovering suitable components, clients can extract this information to assist in component communication. For example the client can extract the events required by the component and a URL to allow the client to subscribe to the services offered by the component. In other words this layer keeps the information necessary to allow a client to communicate with a component.

The Generic and Protocol ontology layers provide the groundwork for the component classification framework. Utilizing concepts stated in these ontologies supports a universal classification of existing components. An UPnP Lamp and a Lamp

using the X.10[3] protocol can be classified as types of the generic *Lamp* described within the Generic layer. Lamp users need now only search for instances of the *Generic:Lamp* component to discover all instances of Lamp within the network, regardless of specific classification. This approach has been further explored within sensor networks, using abstract and generic terms to discover specialized and contextually relevant sensor nodes[6]. Protocols can create logical distinctions between components at the discover layer (for example a Lamp controlled by the X.10 protocol would be distinct from an UPnP Lamp). Using this approach, the protocol of each component now becomes an attribute of the component description. While protocol specific attributes are retained within the component's description, the classification of the component provides a link to a generic classification within the Generic ontology, and therefore with the description framework. The relationships between the concepts (and relevant levels of the ontology stack) are shown in Figure 3. The dashed lines represent inferred attributes of the higher level concepts; so in this example, the UPnP Lamp is inferred to have the category 'Lighting' because of the explicit category relationship specified between the *Generic:Lamp* entity and the *Device:Lighting* entity.

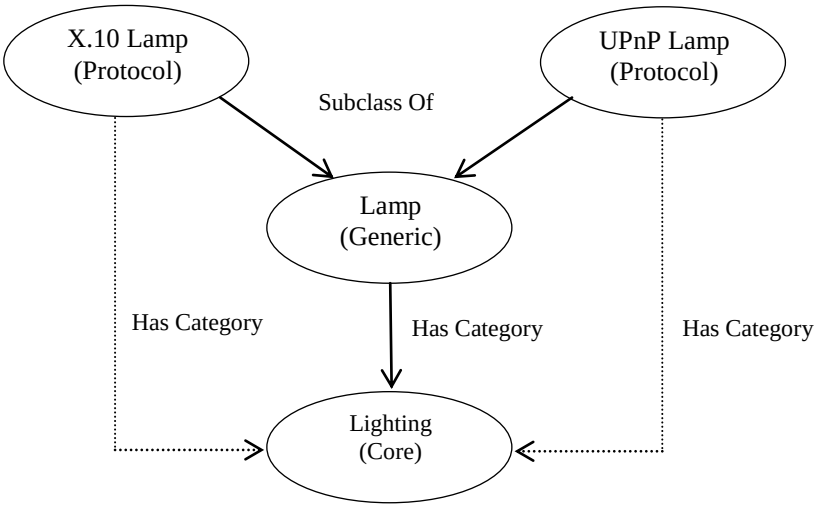

Figure 3. Relationships between Lamp concepts.

This approach can also be applied across a number of middleware platforms. Users are able to retrieve the attributes of available services in a similar manner. So, a Lamp module controlled by a particular service would be classified as a type of generic Lamp. In an environment where multiple instances of *Lamp* occur over different protocols and middleware, a simple search for a generic *Lamp* would return all relevant matches.

In the current implementation the Protocol layer contains ontologies for UPnP and Jini[4] components, in addition to some X.10 devices. This exercises the framework on protocols with descriptions, and protocols that do not have descriptions. So Jini is

[3] A description of the X.10 protocol specification can be found at: http://software.x10.com/pub/manuals/xtdcode.pdf

[4] The Jini specification is now managed by the Apache River project: http://river.apache.org/

included within the Protocol layer as it can be considered a descriptive protocol, from the viewpoint of an external middleware framework.

Both the Generic and Protocol layers are not envisioned to be constrained in size, due to the continual development and deployment of Home Network components. Surrounding the Generic and Protocol ontologies is the domain for externally defined ontologies, which allow developers to reuse terms from all levels of the stack. Exposing these layers allows the range of the stack to be extended as the domain grows. For example, if a new 3D television is introduced to the network, the description of the TV may reuse many of the descriptive terms contained in the ontology stack, such as the services offered by a base television (e.g. *Video display*, *volume control*), or more simple attributes (e.g. *hasPowerRating*). In addition, it may also contain a new service description – *Project 3D Image service*. A generic representation of the service would first be added to the Core and Generic levels of the ontology, so as to provide a minimal generic description of the service. A richer description of the service would then be added to the Protocol level of the ontology stack.

3. Implementation

The Open Standards Gateway Initiative, or OSGi, provides a well defined middleware framework, complete with service life cycle management allowing services to be added and removed. The services are provided in packages called bundles within OSGi. To deploy the approach within OSGi, a Semantic Service Discovery Bundle (SSDB) has been placed within an OSGi framework to provide ontology-based component discovery. The SSDB utilizes open source tools for management of the ontology stack (Jena) and all reasoning and inferencing (Pellet). The SSDB is capable of both dynamic and static description handling. Descriptions can be uploaded from a particular external location, or created dynamically at runtime. This is possible as the information within the SSDB is constantly being reasoned over, and inferred information added where appropriate. For example, a new OSGi service may wish to add a new service description to the framework knowledgebase - e.g. *Project 3D Image*. On adding a description of 3D television and the services offered, the SSDB will be able to absorb the new knowledge about the 3D projection service.

To illustrate the novel unifying approach the testbed exploits OSGi's ability to accommodate a number of protocols using a set of driver bundles. The bundles act as a mediator between the framework and the protocol, typically during service invocation. By adapting driver bundles for UPnP, X.10 and Jini protocols, it has been possible to develop translation components which perform translation of service and device descriptions from their native framework into ontology descriptions grounded in the Ontology Stack vocabulary

As X.10 does not have scope for carrying its own descriptions, some user interaction is needed to set up device addressing. In other words, an intervention is required to create a description of the component. However Jini and UPnP descriptions are translated automatically as a component joins the network.

To facilitate the use of the SSDB by client components, a query interface called the Simple Query Interface (SQI) has been defined. This component is independent from the implementation of the SSDB and allows the testbed to operate independently from the specific discovery functions of the SSDB. This allows the service registry, which provides discovery functionality to the rest of the network, to be replaced with more

suitable management components if required. The SQI also allows querying independently of ontological terms. Queries can be in the form of Simple, Complex (multiple queries over the same component) or Meta Queries which allow sub querying over meta data within the registry. For example, using the SQI a component could request a Lamp service which was located next to the room the user was currently occupying. The SSDB would first infer the desired location, by examining the current location and discovering the suitable location matches for 'next room', and then performs a standard search for suitable components.

Figure 4 shows a high-level view of the system architecture. All translation components use the same instance of the SSDB, allowing the SSDB to have knowledge of all available components within a single ontology model. In the future there is certainly scope for a distributed approach, where an instance of the SSDB is only concerned with one particular protocol, with another component federating requests between them. Here a centric approach is described where there is a single entry point for components, which ensures any applicable new data, inferred or otherwise, is applied to all entities within the ontology model.

4. Translation

A translation component is responsible for translating protocol specific attributes into those contained in either the Protocol or Generic layers of the ontology stack. Some attributes will be protocol specific, and as a result have no grounding within the Generic layer of the stack. For example, UPnP devices may offer a presentation URL for users to interact with, but this property is not common or relevant to other home network protocols. It would therefore be a property relevant only to UPnP descriptions. Other attributes can be translated into generic properties due to their relationship with the protocol specific ontology property. For example, UPnP devices can offer a *serialNumber* attribute. This property is translated into the *UPnP:serialNumber* ontology property described within the UPnP Device ontology. Within this ontology, *UPnP:serialNumber* is declared as a sub-property of *Device:serialNumber* described within the Device Ontology. This ontology lies within the Generic layer of the stack, and therefore a relationship between a protocol specific attribute and a more generic attribute is formed.

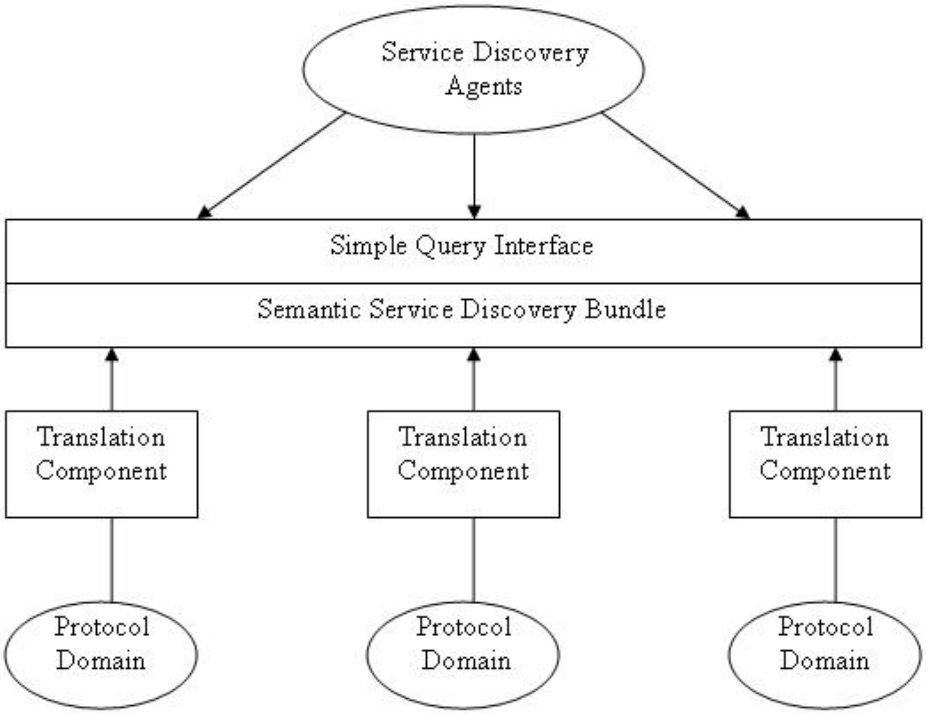

Figure 4. System Architecture.

Translation components also contain a list of associations between both protocol specific services and device categories, and their "peers" within the particular protocol domain ontology. For example, the X.10 translation components knows in advance that an X.10 Lamp is an instance of the *X10:Lamp* device described within the X.10 ontology. Initially, this method may seem too static for a potentially dynamic environment. However the OSGi framework resolves this issue, as components can be updated after deployment, allowing for new associations to be added as new devices and services are developed. On retrieving an association, the translation component adds a new instance of the ontology device or service to the SSDB, complete with translated attributes. Reasoning and inference can now take place.

The SSDB manages the whole ontology stack, adding new information as it becomes available. As a new translation component joins the network, the new protocol ontology is uploaded to the SSDB and added to the stack. Any developer-defined ontologies provided by network components are also added onto the stack. This allows the SSDB to perform inference and reasoning over all available information, providing an accurate discovery environment.

Each translation component only has knowledge of its particular domain ontology, and its relation to the lower levels of the stack. For example, the UPnP translation component may have knowledge of the Base, Core and Generic level ontologies, but its knowledge of the Protocol level is limited to the UPnP ontology.

5. Domain Application

This section describes implemented examples of the SSDB in use. The first example covers the description framework being applied to multiple protocol domains, and highlights how the approach simplifies component classification and discovery. This is followed by a second example, which discusses the use of the approach within the MATCH [4] project, a home-care system underpinned by Home Network technology.

5.1. Classification in a multi-protocol domain

Earlier the chapter describes how driver components are augmented to create translation components for the SSDB. Many Home Network projects describe examples involving unique services, such as codec translation services [7] and user availability services [3, 8]. Such services are usually limited to a single protocol, that is to say examples of these services are not commonly found across a number of protocols. In order to demonstrate the usefulness of the SSDB, it was necessary to implement an example using lamp devices across three protocols. X.10 is a protocol that operates along power cables such as those found in a home, and is inherently extremely suitable for use with existing lamps. The UPnP protocol also has schemas for describing light devices. However there are no such well defined interfaces for Jini, so a *LampInterface* class simulating such a component has been created for the testbed.

Hence there are three unique instances of Lamp components within the Home Network; one for each protocol. While these three components are within their own protocol domain, the translation components allow us to interpret their specific characteristics into terms relative to the SSDB and Ontology Stack. The X.10 translation component is a simple user interface, which takes input from the user. The UPnP translation component is built upon an UPnP control point, which retrieves descriptions on behalf of the component. The Jini translation component acts as a Jini service listener within the Jini network. A 'listener' component acts on behalf of a protocol with respect to service discovery within the home network as a whole. It may offer services found within the home network to components within its own protocol domain, and vice versa. On connection to the network, the translation component for each protocol interprets and classifies the attributes of the particular lamp into terms within the Ontology Stack. For example, the UPnP Lamp is denoted as being an instance of the *UPnP:Lamp* concept (which, as shown in Figure 3 as a subclass of *Generic:Lamp*).

As a result there are three instances of *Generic:Lamp* components available to the network. When a client component submits a SimpleQuery through the SQI to the SSDB to find instances of *Generic:Lamp* components; it is returned each component as a suitable match. (Remember driver components within the OSGi framework are used to start services across protocol boundaries.) The client chooses which device it wishes to use and queries the SSDB to obtain the protocol specific details, such as the device address which is presented in a protocol-specific manner. How a client selects a service or device to use is protocol specific, and is typically handled by the corresponding listening/translation component. The client then passes these details to an agent responsible for carrying out the action. How this action is executed is dependent on the protocol, but as the Ontology Stack contains a layer for protocols, the SSDB retains the protocol specific information along with that related to the Generic layer. Middleware is used to blur distinctions between protocols at the invocation layer.

What becomes clear through this example is, by using the Ontology Stack and SSDB, it is also possible to blur the distinction between protocols at the discovery layer.

A broader advantage of the approach is the ability to handle complex queries that may include for example positional information. This is an aspect that is particularly pertinent within the home. This ability emphasizes the value of the SQI. Consider a MetaQuery asking for a light device within the room opposite the living room. To do so, first form a SimpleQuery which discovers the room opposite, and then use this object within the MetaQuery. At an abstract syntax level, the query looks like:

MetaQuery (*Generic:Lamp*, *Device:locatedIn*,
SimpleQuery (*Location:Room*, *Location:opposite*, *Location:LivingRoom*))

Using such queries reduces the number of interactions required with the SSDB. This also has the useful property of allowing components to query using information they do not immediately know.

5.2. The SSDB within the MATCH system

Earlier chapters explain the reach of the MATCH project. In short it is a research project concerned with using Home Network technologies to provide care and support for those at home. This aim is achieved in part through the use of existing simple devices and technologies to derive high-level information, feedback, and monitoring for stakeholders. This section briefly describes the impact of this service discovery approach within the MATCH architecture.

The MATCH system [4] is built upon the OSGi framework, but communicates by means of a message broker system, allowing the system to abstract the communication from the particular platform used. This requires an extra layer to be placed upon component descriptions to provide details of how components can operate together. MATCH components own channels through which users interact with the component, and a set of ontologies have been developed to describe the metadata associated with this communication system. Where possible, ontology terms within the Ontology Stack have been reused within the MATCH ontologies, grounding these new ontologies within the established domain. For example, MATCH components have locations of type *Base:Location* and *Location:Room*. Similarly, components can be attributed to services and devices already described within the Core and Generic layers of the stack. A MATCH-Touch Screen Interface would inherit many of the service descriptions offered by the generic *Visual Device* and *Pointing Device* devices found within the Generic level of the stack. While many of the functional aspects are similar, the communication takes place using the message broker system and so the communication descriptive properties are specific to MATCH components.

This means in essence that the ontologies describing MATCH components become another entry within the Protocol layer of the Ontology Stack, with instances of these descriptions lying within the User Defined layer. The MATCH project highlights the extensive capabilities of the approach, by allowing emerging protocols to use their own description frameworks, while also grounding these within a well-defined description framework.

The SSDB also acts as a repository of knowledge for the home care network. A room-occupation software component will make use of the ontology registry to discover all locations in the home which have room or occupation sensors.

Stakeholders can use the data from the software component to determine the locations used the most by the home occupier. On discovering common behaviour patterns or the most popular locations, stakeholders can query the registry further to determine which devices or services are present in those locations, and so investigate possible links between occupation and activity.

6. Alternative approaches

Use of ontology languages within the computing domain is becoming more popular. Logical properties, coupled with the potential for rich descriptions, have encouraged researchers to apply ontology languages to conventional network domains.

6.1. Ontologies and Web Networks

A wealth of ontology-based projects can be found for web services, where semantically-rich descriptions extend the basic WSDL[5] documentation of a service. WSDL stands for *Web Services Description Language* and essentially a WSDL description is a document written in a structured format (XML) which describes a Web service. It specifies the location of the service, and the operations and methods the service offers.

Web services contain many similarities to a Home Network environment, as both have services that inter-operate independently of their internal platform or protocol. Two major projects are currently developing schemas and vocabularies employing ontology languages: one is focused on the use of a language called OWL-S which based on OWL and was mentioned earlier; the other utilizes the Semantic Web Service Framework[6] (SWSF). Both syntaxes allow developers to describe their services in such a way as to provide a rich description of what the service is, how the service operates, and how users can interact with it. In other words, the languages provide meaning or semantics to a description. Although these languages are popular within the research community [9, 10], their syntax is still under development. As the syntax may change in future revisions, these languages are not considered appropriate for the Home Network domain. In contrast the OWL syntax became a W3C recommendation in 2004. The World Wide Web Consortium (W3C) is an international community where member organizations supported by a full-time staff, work with the public to develop Web standards.

6.2. Ontologies and Middleware Frameworks

As ontology languages have matured, several ontology-based projects in the domain of home and P2P networking have emerged, and as a result useful development and management tools have appeared.

GloServ [11] is a project which utilizes the descriptive characteristics of ontologies to provide a hierarchical approach to describing and locating services across peer-to-

[5] The Web Service Description Language (WSDL) is a W3C specification for the use of services over networks: http://www.w3.org/TR/wsdl

[6] The Semantic Web Service Framework is a current submission to the W3C organisation: http://www.w3.org/Submission/SWSF/

peer networks. Ontologies are used to provide a structured mechanism for locating registry servers based on service categories and properties. The project is concerned with providing a taxonomy of services that are known to service registries, and by relating the location of service registries to each other. Rather than providing detailed descriptions of services, it is orientated towards discovering domain information about the service registries, determining domain hierarchies, and locations.

Common approaches to embedding knowledge into middleware frameworks [9, 12, 13] utilize ontology languages to provide logical and semantically rich component descriptions in the framework. While this is undoubtedly a step forward in assisting frameworks to interpret and infer, rather than just extract component attributes, the vast majority of these approaches require participating services and components to adhere to the framework's ontology-based vocabulary. In addition to exacerbating discovery issues within multi-protocol environments in general, this approach also excludes the majority of existing Home Network components and protocols. Some existing protocols, such as UPnP, are attractive candidates for applying to an ontology-based Home Network description framework, due to their existing description framework or schema. Similarly, existing middleware frameworks typically only contain their own description framework for their particular domain.

Ontologies have also been used to support context aware systems [8, 13]. In such systems, the rich expressiveness of the language is exploited to derive high-level contextual information from low-level data. High-level states, such as room occupancy or user availability, are described using restrictions on low-level sensor data. For example, a meeting room is full if there are more than five people currently in it. The Gaia [14] infrastructure takes this reasoning to a higher level, allowing fuzzy logic to be used where exact states cannot be fully inferred.

The Network Appliance Service Utilization Framework [7, 15] (NASUF) is one of the most promising ontology based Home Network projects. In this framework, services are described using an ontology-based vocabulary which is designed to support not only simple service description, but also dynamic service composition [12]. This description framework leverages the syntax developed within OWL-S. The NASUF also exploits the logical properties which OWL provides, reasoning over descriptions in an attempt to provide a best match. This framework has also been applied to a P2P environment [7].

The approach described in this chapter differs as it is concerned with reusing existing description frameworks, rather than defining anew and requiring all participating components to adhere to this new definition. In this manner, existing technology can be integrated into an existing Home Network environment. In other words the approach seeks to exploit and use existing descriptions and effort, and allow such isolated work to be combined into a greater whole. The approach is to provide a means of cooperative working rather than constantly repeating effort and starting afresh each time.

7. Motivation and comparisons

The use of ontologies within networks is becoming a popular research area, but this creates two main issues which remain to be tackled. Firstly, the issue of middleware 'lock-in' hampers the true interoperability of middleware frameworks at the description layer, as existing components are ignored and those in development tend to be domain

or framework specific. Secondly, the lack of a standardized classification taxonomy limits the discovery process over multiple domains.

The issue of 'lock-in' can be addressed using the logical properties of OWL which allow concepts from one ontology to be declared equivalent to those within another ontology, providing that no logical conflicts occur. That is, creating 'meta ontologies' to describe the relationships between two concepts from distinct ontologies can unify existing Home Network ontologies. In domains where much similarity exists, search agents developed for either ontology would be able to operate over both sets of ontology data, expanding their search domain. This would not require a modification to the search agent. If the agent was looking for Device concepts from ontology A and the Device concept from ontology B had been declared equivalent, the agent would view a device from ontology B as if it were from its own domain. Hence independently developed ontologies can be reconciled with each other, and so alleviate 'lock in' and expand the domain for the search agents. External ontologies can be incorporated into the Protocol level of the Ontology Stack, acting as if they are simply a new protocol, with a mapping ontology describing the relationships between the external and internal vocabulary.

The Generic layer of the Ontology Stack provides a solution to the second issue of multiple domains by incorporating any existing multiple Home Network protocols within the component discovery process. Using concepts and vocabulary from the Generic layer allows classification and inference against generic and common terms across multiple domains. The Generic layer provides a simplified taxonomy for common descriptions while the higher layers of the stack provide a framework which ensures a rich descriptive vocabulary, where terms are tightly coupled to their protocol-specific counterparts.

Frequently, ontology-based projects use their own specific ontologies for describing network components. While the approach of applying ontologies to this domain is novel, this ultimately leads to the middleware framework requiring all components to be able to describe themselves in a way which the framework understands. Due to the compact and pre-programmed nature of services this is not always possible, and certainly rules out the vast majority of existing Home Network components.

A popular approach to ontology based discovery [7, 15] is concerned with the inputs, outputs, pre-conditions and post-conditions of a service. The reasoning behind this principle is that if a client service knows what the service provider is expecting in terms of service input, and what to expect as service output, then services can interoperate without binding to a predefined service interface. This approach is slightly misguided. In order for true discovery to take place, users need to know what service or device they require before discovering how to interact with it. In existing ontology based frameworks, the classification of a service or device is largely a simple value, and assumed to be a reference to an externally defined taxonomy.

The need for such a taxonomy has motivated researchers to develop specific ontologies to support the classification of Home Network components. The Foundation of Intelligent Physical Agents [16] is one project of note which has developed a substantial ontology specification, to aid the experience of intelligent agents. Other approaches [17] concentrate on a more generic domain of device and service ontologies to increase the suitability of the descriptive framework. In these approaches however, the domain has grown rather large and rather too generic; and indeed to be useful to a

specific environment such as a home network, a large amount of the terminology would become redundant.

Despite such efforts, researchers and developers continue to develop description ontologies independently of those already in existence. It is the opinion of the authors that the lack of a standardized ontology is due to a lack of agreement amongst peers as to what should be in the ontology, rather than the lack of suitable existing ontologies. Indeed, it is also difficult to specify a complete ontology while the domain is continually developing.

Using the authors' approach, it is possible to resolve these issues, and ultimately present ontologies as a valuable contributor, not only to the Home Network, but to other service orientated networks such as Grid Services.

8. Looking forward

The main contribution of this chapter has been to focus on the integration of protocol descriptions into the Home Network Ontology Stack and the suitability of the vocabulary for the environment; all of which has highlighted by work on the MATCH system. In order to flesh out the protocol layer of the Ontology Stack it will be necessary to examine other potentially suitable candidates to incorporate, such as the HAVi and Zigbee architectures. Difficulties may occur where protocols do not include a standard set of descriptive terms although, as demonstrated with X.10, this does not necessarily exclude such protocols from this new approach. This approach to service and device classification can be applied to domains other than the Home Network, such as web services. Rather than replace existing ontology-based description languages, such as OWL-S, the approach can enhance the overall discovery process. It is also necessary to examine the suitability of the approach in enhancing the initial discovery process, where agents can utilize the ontology stack for initial classification of services, and then exploit existing web service description languages for service usage.

References

[1] D. Marples, & P. Kriens, 'The Open Services Gateway Initiative: An Introductory Review', *IEEE Communications Magazine*, **39**(12) (2001), 110-114.

[2] C. Lee, D. Nordstedt, & S. Helal, 'Enabling Smart Spaces with OSGi', *IEEE Pervasive Computing*, **2**(3) (2003), 89-94.

[3] N. Georgantas, S.B. Mokhtar, Y.-D. Bromberg, V. Issarny, J. Kalaoja, J. Kantarovitch, A. Gerodolle, & R. Mevissen, '*The Amigo Service Architecture for the Open Networked Home Environment*', WICSA '05: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, 2005, 295-296.

[4] P. D. Gray, T. McBryan, N. Hine, C. J. Martin, N. Gil, M. Wolters, N. Mayo, K. J. Turner, L. S. Docherty, F. Wang, & M. Kolberg '*A Scalable Home Care System Infrastructure Supporting Domiciliary Care*' Technical Report CSM-173, University of Stirling, August 2007.

[5] Gray, A.J.G.; Sadler, J.; Kit, O.; Kyzirakos, K.; Karpathiotakis, M.; Calbimonte, J.-P.; Page, K.; García-Castro, R.; Frazer, A.; Galpin, I.; Fernandes, A.A.A.; Paton, N.W.; Corcho, O.; Koubarakis, M.; Roure, D.D.; Martinez, K.; Gómez-Pérez, A. 'A Semantic Sensor Web for Environmental Decision Support Applications', *Sensors* 2011, 11, 8855-8887

[6] Siming Yang, Yang Xu & Qingyi He, 'Ontology Based Service Discovery Method for Internet of Things', *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, 2011, 43-47.

[7] P. Fergus, M. Merabti, M.B Hanneghan, A. Taleb-Bendiab, & A Mingkhwan, '*A Semantic Framework for Self-Adaptive Network Applicances*' IEEE Consumer Communications and Networking Conference, Las Vegas, USA, 2005, 229-234.

[8] X. Wang, J.S. Dong, C. Y. Chin, S. R. Hettiarachchi, D. Zhang, 'Semantic Space: An infrastructure for smart spaces', *IEEE Pervasive Computing Magazine*, **3(**3) (2004), 32-39.
[9] B. Falchuk, D. Marples, 'Ontology and application to improve dynamic bindings in mobile distributed systems' *Proceedings of the 2nd annual international workshop on Wireless internet*, Boston, Massachusetts, Article No. 23, 2006.
[10] E. Sirin, B. Parsia, & J. Hendler, '*Filtering and Selecting Semantic Web Services with Interactive Composition Techniques*' IEEE Intelligent Systems, 19(4) (2004), 42-49.
[11] K. Arabshian, H. Schulzrinne, D. Trossen, & D. Pavel, 'GloServ: Global Service Discovery using the OWL Web Ontology Language', *The IEE International Workshop on Intelligent Environments*, Colchester, England, 2005.
[12] A. Mingkhwan, P. Fergus, O. Abuelma'atti, M. Merabti, R. Askwith, & M. Hanneghan, '*Dynamic Service Composition in Home Appliance Networks, Mutimedia Tools and Applications*', A Special Issue on Advances in Consumer Communications and Networking, **29**(3) (2006), 257-284.
[13] P. Pawar, & A. Tokmakoff, 'Ontology-Based Context-Aware Service Discovery for Pervasive Environments', *1st IEEE International Workshop on Services Integration in Pervasive Environments SIPE*, June 2006.
[14] A. Ranganathan, J. Al-Muhtadi, & R.H. Campbell, 'Reasoning about Uncertain Contexts in Pervasive Computing Environments', *IEEE Pervasive Computing*, **3**(2) (2004), 62-70.
[15] A. Mingkhwan, P. Fergus, O. Abuelma'atti, M. Merabti, 'Implicit Functionality: Dynamic Services Composition for Home Networked Applicances' *IEEE International Conference on Communications*, **1**, June 2004, 43-47.
[16] FIPA Device Ontology Specification, Foundation for Intelligent Physical Agents (FIPA), Geneva, Switzerland, 2002. http://www.fipa.org/specs/fipa00091/XC00091C.pdf .
[17] A. Bandara, T. Payne, D. de Roure, & G. Clemo, 'An Ontological Framework for Semantic Description of Devices', *The 3rd International Semantic Web Conference*, Japan, November 2004.