INVITED PAPER

Managing Distributed Networked Appliances in Home Networks

Early experimentation indicates that it may be feasible to connect ordinary household appliances into home computer networks for self-configuration and self-management.

By Madjid Merabti, Paul Fergus, Omar Abuelma'atti, Heather Yu, and Charlie Judice, *Fellow IEEE*

ABSTRACT | Recent years have seen an exponential growth in the use of home networks, from the Internet-enabled PC to network-enabled home appliances. Ordinary and everyday appliances used in the home will increasingly become integral components of these networks. They are required to join, leave, and self-configure in accordance with their dynamic environment. New platforms and applications are needed to mediate interactions between devices to overcome the inherent problems, limitations, and costs of bespoke solutions. While combining our disparate devices to meet future needs is challenging, it will also allow for better exploitation of networked devices to provide obvious benefits to the consumer. A number of approaches and technologies that attempt to redress this issue exist; however, they are not without their own disadvantages. This paper discusses these technologies and introduces some of the novel approaches that have been proposed to address the issue of device autoconfiguration using peer-to-peer technologies. In this way, disparate devices are able to achieve self-management without user interaction. A case study is also presented to illuminate these approaches and demonstrate the applicability of our approach.

KEYWORDS | Dynamic service composition; interoperability; multimedia services; networked appliances; self-adaptive middleware

Manuscript received October 13, 2006; revised April 4, 2007.

M. Merabti, P. Fergus, and **O. Abuelma'atti** are with the School of Computing and Mathematical Sciences, Liverpool John Moores University, L3 3AF Liverpool, U.K. (e-mail: M.Merabti@ljmu.ac.uk; P.Fergus@ljmu.ac.uk; O.E.Abuelma'atti@ljmu.ac.uk).
H. Yu is with the Technologies Laboratory, Panasonic Information and Networking, Princeton, NJ 08540 USA (e-mail: heathery@research.panasonic.com).
C. Judice is with the School of Science, Technology and Engineering, Monmouth University, West Long Branch, NJ 07764 USA (e-mail: cjudice@monmouth.edu).

Digital Object Identifier: 10.1109/JPROC.2007.909922

I. INTRODUCTION

Networked devices have revolutionized the way we access and disseminate information and changed the way we communicate with each other. More and more homes are now Internet-enabled as people from all walks of life embrace this technology and the benefits it brings. Such benefits include shared Internet access, file and peripheral sharing, voice over IP, and online gaming amongst others. As this trend continues, we can expect ordinary everyday appliances to become part of these networks, with networked devices becoming ever more pervasive and often invisible to users. At the simplest level, this might mean lights that switch themselves on when someone enters the room, or an Internet radio appliance. Looking further ahead, home appliances will be required to automatically integrate, configure, and manage themselves, and to form high-level applications. For example, garden sprinklers may be networked to form close relationships with humidity sensors, real-time weather readings and weather forecasts received via the Internet to make decisions on whether sprinklers should be turned on or off, thus optimizing and reducing water consumption and costs.

Researchers suggest that this transition mirrors the evolutionary process undertaken within personal computing and wide-area communications more generally; it is now difficult to imagine using a computer without Internet access. Given the success of this transition, home networking platforms aim to achieve the same level of acceptance. Fig. 1 illustrates how such an interconnected home environment might look. Here we see the home supporting many different devices, each equipped with different communication capabilities such as broadcast, mobile, and the Internet. The challenge is to create ways to

Fig. 1. *Connecting home appliances.*

easily connect all of these devices together where they can be automatically managed with little or no human involvement and in such a way that users need not worry about how they are connected together.

Several definitions for networked devices exist; consequently, it is difficult to provide a clear and decisive description of their key characteristics. From a hardware perspective, Moyer *et al.* [1] define networked appliances as "a dedicated function consumer device with an embedded processor and a network connection." This definition can be extended to include any hardware device that has networking capabilities and the ability to disperse its capabilities within the network allowing other devices to combine and use them to solve some given task. Distributing device functionality in this way allows them to be controlled, monitored, managed and extended beyond what they were initially designed to do. Addressing this challenge is difficult. Solutions have already been proposed to describe and discover network resources using simple keyword matching [2], [3], however these are typically oversimplified solutions and many researchers believe more intelligent mechanisms are required for an effective solution [4], [5]. Although many solutions exist one that has particular potential is the use of ontologies [6]–[8] to help devices describe themselves and the operations they provide.

Given the widespread adoption of networked appliances, new and novel platforms (a framework on which hardware and/or software can run) will be required to overcome the inherent problems associated with bespoke and proprietary development. Such platforms will allow networked appliances to be more easily combined to meet the unique needs of the user and the environment. Achieving this will ensure users are not burdened with the details of this process. For example, a user will have the ability to introduce new speakers into the home that automatically and wirelessly communicate and configure themselves with the home stereo without human intervention. In this way, the integration and management of these devices is achieved. This paper considers the difficulties and possible solutions for achieving this.

II. NETWORKED APPLIANCES AND HOME NETWORKING

The home has been the focus of many research initiatives for several years, and a great deal can be learnt from the existing work in this area.

One such initiative is that carried out by the Massachusetts Institute of Technology, called the Intelligent Room Project [9], [10], where computational functionality is embedded within the environment to interpret real-world human interactions. Using a layered architecture, computation is readily available to the user without having to change his mode of thinking. At the lowest level, perceptual systems interpret what is happening in the room using tracking, pointing, and voice-control mechanisms. Depending on what is perceived, lights can be switched on or off, video can be recorded or played, and images can be displayed. Perceptual systems are connected, via interoperability agents, to high-level application agents allowing the intelligent room to perceive human interaction and execute tasks. The primary focus of the Intelligent Room Project is an experiment in human–computer interaction rather than the networking and automatic interoperation of devices.

Taking an opposing view to that of the Intelligent Room Project, a number of home networking solutions have been developed to try to achieve seamless interoperability. This has included the introduction of a wide spectrum of wired and wireless infrastructures and network protocols such as LonWorks, CEBus, SmartHouse, VHN, HomePHA, HomePnP, IEEE1394 (Firewire), X-10, IrDA, IEEE802.11b, Bluetooth, and HyperLAN/2 [11]. Despite the long list of advantages they provide, several challenges need to be considered, most notably, interoperability [12], [13] and the difficulties associated with the integration of combined functionalities, including the use of vocabularies to discover and describe devices and their operational capacity.

Many other industry efforts have also evolved to create interworking solutions, including the Home Electronic System [14], Home Audio-Video Interoperability [15], and Intel Digital Home.[1] Furthermore, the provision to monitor and control the home using TV sets and set-top boxes has advanced rapidly in recent years, with the TV being considered as the central appliance within a typical home environment [16]–[18]. While such approaches have produced significant results that are already available, they do not provide any capability for automatically composing and managing functionality between and across connected devices. They therefore fail to capitalize on the full benefits and flexibility that the networking of devices promises.

Looking at specific examples of interoperability, the ePerSpace project[2] aims to provide an end-to-end solution

[1]http://www.intel.com/technology/digitalhome/.
[2]France Telecom, http://www.ist-eperspace.org/.

for value-added audiovisual services. Services are distributed via open access networks based on the details defined in personalization profiles. Using a trusted and interoperable framework, devices are interconnected, allowing them to be controlled by content creators using rich media object management tools. This allows content and devices to be dynamically adapted to specific users. The content-oriented approach taken by ePerSpace is important but largely orthogonal to the development of enabling platforms for interoperation, and it could be integrated into other schemes such as that presented in this paper to form part of a much larger strategy aimed at addressing all interoperability problems.

Taking a broader view, the Digital Living Network Alliance (DLNA)[3] suggests that interoperability must exist between devices that reside within three domains: the Internet, broadcast, and mobile. DLNA advocates that the key to successful integration is to address customer demands where the devices they own work together within and across these domains. From a technical perspective, DLNA argues that this requires design choices constrained through industry consensus to enable better interoperability. They state that current open standards are too flexible and, consequently, interoperability between different vendors fails. Given this limitation, standards in conjunction with proprietary manufacturing are proposed to reduce the time taken to deliver products to high-street stores. Like ePerSpace, DLNA only addresses basic interoperability. Their approach could be incorporated into a larger framework to provide a base solution. However, mechanisms are still required to effectively manage devices.

The Open Services Gateway Initiative (OSGi)[4] assumes network-enabled devices and in-home networks already exist. Using set-top box technologies, networked appliances are interconnected and controlled. This approach has considerable industrial and academic backing from device manufacturers and service providers alike. Its mission is to create open specifications that enable the delivery of multiple services over wide-area networks (WANs) to home networks. The framework incorporates three entities: the service and network provider; the services gateway, and the in-home network. Service providers enable the provision of value-added services to residential customers via the services gateway. Service operators manage and maintain the services gateway and its services, while network providers offer the necessary network infrastructure to enable communication between the services gateway, the gateway operator, and the service provider. The OSGi architecture adopts a centralized approach providing little support for ad hoc environments. Federated and distributed registries can be used to partially address this limitation; however, they can be difficult to extend and maintain, they have increased message load, and registry entries may be unreliable or inconsistent.

Unlike the limiting set-top box characteristics of OSGi, the Reconfigurable Ubiquitous Networked Embedded Systems (RUNES) [19] project is trying to provide a platform that allows a larger number of devices to form part of the Internet. To support such environments, it aims to control how devices are interconnected and operated, thus providing users, designers, and programmers with the flexibility to interact with services, devices, and sensors. It promises to deliver a platform that supports large-scale distributed, heterogeneous, networked systems that can interoperate and dynamically self-adapt to environmental changes.

Nonetheless, a number of questions can be raised about its larger interoperability strategy. Well-defined language-independent component metamodels are bound to service interfaces allowing interoperation and self-management of components through the composition of these metamodels. This constrains how applications are developed and a more flexible approach might allow different device manufacturers to describe metamodels and interfaces using their own preferred vocabulary. The use of semantics is discussed in more detail in the following section, with many approaches having been proposed.

While solutions may move towards service-oriented architectures, there still remains a problem with how such services will be deployed and managed. OSGi argues that centralization is the best way. However, ad hoc networking standards are beginning to highlight the benefits peer-to-peer technologies might bring. In these networks, devices are seen as self-contained nodes, where functionality is received from within a device's immediate vicinity. The AMIDEN [20] project builds on this concept by allowing devices to disperse the functions they provide as independent services. A standard with considerable support that also adopts this notion is Universal Plug and Play (UPnP).[5] Devices are automatically discovered, interconnected, and controlled within local-area networks, thus extending the autoconfiguration features of device Plug and Play (PnP). It achieves interoperability by leveraging existing mature standards currently used by the Internet and Web such as IP, HTTP, and XML. It provides mechanisms that can use existing addressing schemes such as DHCP and AutoIP functions best suited to ad hoc networking. The main drawback of UPnP is that it cannot be extended to wide-area networks due to the networking techniques it uses. This is a significant limitation that might seriously threaten the standard, especially as, in modern networked environments, the divide between local- and wide-area networks will become increasingly blurred as more and more capabilities are shared between the two.

Although peer-to-peer networking appears to have reinvented itself, it has been plagued by technical

[3]http://www.dlna.org/about/DLNA_Overview.pdf.
[4]http://www.osgi.org/.
[5]http://www.upnp.org/.

difficulties that still make this approach problematic. Dating back to Napster, peer-to-peer has adopted numerous techniques ranging from unstructured topology construction—as in Gnutella [21]—to more structured approaches as demonstrated in Proof [22]. It is difficult to overcome all limitations inherent in these approaches. For example, network flooding can alleviate the need to continually update distributed hash tables, but only at the expense of increasing network traffic. Taking an opposing view, network traffic could be reduced using structured peer-to-peer networks, but this would increase the effort required to maintain network consistency. This is because hash tables need to be continually updated to reflect network state. Given these conflicting challenges, no single approach has been developed to overcome both of these issues.

Nonetheless, peer-to-peer is becoming increasingly important within networked appliance research where there is a need to disperse operational capacity and utilize these functions in ad hoc environments [23], [24]. A new specification that promises to provide a platform to achieve this is JXTA [25]. JXTA leverages the benefits of distributed hash tables and extends the benefits of UPnP to help devices expose their capabilities as services, allowing them to be reused within the network. JXTA overcomes the limitations discussed above by creating a hybrid system that uses a loosely consistent distributed hash table [26]. The advantage with this approach is that it is less expensive to maintain. The disadvantage is that it may be temporarily or permanently inconsistent.

JXTA, like many other standards, provides a mechanism for interconnecting devices. It also addresses an important requirement: that of exposing device functionality as services. However, JXTA fails to provide mechanisms to automatically manage devices and the services they provide. This is seen as a key requirement within the discussion presented in this paper where devices are not just interconnected but also managed. So while standards such as JXTA can underpin an effective approach, we must extend its capabilities to incorporate advances made—for example—in the semantic Web services community. This allows devices to proactively form relationships with each other and automatically manage interactions, as we shall see in the next section.

III. SEMANTICS, SERVICES, AND DEVICE COMPOSITION

While many definitions of ontology exist, perhaps the most useful is that provided by Uschold *et al.* [27] as "a shared understanding of some subject area which helps people or processes achieve better communication, interoperability and effective reuse. The ontology embodies a conceptualization—definitions of entities, their attributes and relationships that exist in some domain of interest. The conceptualization is explicitly represented." The formal serialization of concepts allows functionality to be described in a way that promotes machine-to-machine interoperation, something that could prove conducive to networked appliance and home networking research.

The semantic Web services community is trying to address this requirement by utilizing ontological metadata to dynamically deploy and compose services within WANs. Sycara *et al.* [28] describe how the Web ontology language for services (OWL-S) can be used to describe services in terms of their capabilities. Building on this approach, Paolucci *et al.* [29], [30] have developed a matching engine that determines the similarity between a service request and the description of a service by evaluating the inputs a service takes, the outputs it generates, the preconditions that need to be satisfied, and the effects that happen as a result of executing the service. They use a term called "sufficiently similar"; in its strongest sense, a service description and a service request are sufficiently similar when they describe exactly the same service. This, however, is too restrictive, because advertisers and requesters have no prior agreements on how a service is presented. To accommodate a softer definition of "sufficiently similar," Paolucci *et al.* perform matches based on the degree of similarity between the service request and the service description.

While this approach is interesting, it only performs matches using the *service profile* model. It does not process the remaining service ontologies defined in the OWL-S specification to determine if the information provided in the service request can be directly mapped onto service bindings described in the service interface. Addressing this limitation will become increasingly important as we move away from semiautomated service composition (user-centric) towards approaches where devices automatically perform these composition activities themselves.

Nonetheless, early results suggest that semantics may provide a line of enquiry worth pursuing. This is evident in SWORD [31], where service compositions are generated using semantic descriptions. Ponnekanti *et al.* from SWORD claim this has the advantage of providing specific advice at multiple levels of granularity during the service composition process. At the highest level, the system can help determine which kind of service is required against a contextual backdrop that indicates problem-solving goals and procedural knowledge. In-depth advice can be given, for example, on how to initialize and configure the control parameters of a service. Users describe the service they require using pre- and postconditions using first-order logic. Typically, this is performed by experienced users or domain experts [32] and would not be appropriate for novice users. The challenge is to remove the need to burden the user with potentially mundane device and service integration tasks. This is seen as an important requirement, as devices and services become more numerous. Consequently, users will simply not know how to combine devices to perform tasks that are more complex in nature.

An approach that adopts similar techniques is Component Service Model with Semantics (CoSMoS) [33]. Using semantic graphs, ontologies, and object-oriented technologies, CoSMoS provides an intuitive way of describing high-level concepts in a more flexible manner. Users assemble services with natural language phrases, aided by semantic graphs and ontologies, a complex process due to the difficulties associated with natural language processing [34]. Using concepts to describe services, compositions are created by forming explicit links between these concepts. Like SWORD, CoSMoS is user-centric. No mechanisms are provided to automatically compose services, and its querying capabilities are restricted to simple scenarios.

Combining semantic Web services technology with pervasive computing, Fujitsu Laboratories has coined the term "task computing," where a user-oriented framework allows nonexperts to create complex tasks using applications, devices, and services [6], [7]. Services are used to abstract device functionality described using OWL-S and discovered using Universal Plug and Play. Using the Semantic Task Execution Editor (STEER), users can dynamically publish and discover services, create tasks, and execute those tasks on-the-fly [35], [36]. Compositions are created using the principles of semiautomatic composition as defined by Sirin *et al.* [37], where user-selected services are composed to provide some value-added service.

Task computing provides an interesting user-oriented approach much like the approaches described above. Although the user is obviously important, the common objective is to avoid overtaxing the user with unrealistic demands. While small tasks may be easy to create, this may not be the case as tasks become more complicated. As such, it is important to underpin task computing with automated functionality that not only aids task creation but also provides autonomic principles to help tasks adapt.

An approach similar to that of task computing is *appliance aggregation* [38], [39]. This approach adopts four levels of aggregation, where the lowest level provides shared ownership of appliances and the highest provides access to the same functionality, which is seen as the most controlled layer. Using these levels of aggregation gives the illusion that multiple appliances appear as a single device for the duration they are composed. At the highest level, services are automatically aggregated using user preferences and historical data collected from user and device interactions. Initially, aggregations are created based on existing descriptions of devices and user requests. This is achieved using best match principles, which can be changed or corrected by the user.

The approach is user-centric, with possible solutions being presented to the user based on historical data and predefined policies. User interactions are oversimplified where complexity is set to increase and it is not clear whether appliance aggregation could accommodate this complexity. While dynamic environments are used, there is little evidence to suggest that devices and services could be automatically composed through machine-to-machine negotiations, devoid of any human intervention.

Moving away from user-centric service composition techniques, Chakraborty *et al.* [40] argue that a better approach is to form compositions and execution paths based on the principles of reactive systems. Using their reactive architecture, service compositions are only executed upon request. They describe request planning as the creation of process models, which compose multiple services to satisfy a particular query. This is typically known as workflow management. The execution of the composite request is the process of discovering, integrating, and executing multiple services in a planned order. This is achieved using a distributed broker that can be executed on any node in the ad hoc environment. The broker coordinates the discovery, integration, and execution of different services to calculate the results for a composite request. An individual broker handles each composite service request in the environment. The composition architecture deals with the execution of a composite request and assumes that the process model would be provided to it. Although "reactive" appears to be a sensible approach, it is not clear whether the work will be extended to dynamically create compositions on the fly.

Based on our findings, it is clear that automatically composing services is a difficult problem where many approaches prefer to include the user or predefined workflows. Fujii *et al.* [33] argue that despite the difficulties associated with dynamic service composition, it provides several benefits, most notably flexibility, adaptability, and availability. They describe how flexibility enables a number of different applications to be composed from a set of available components. Dynamically composing services in this way allows applications to be composed according to the user's preferences and context. Perhaps just as important, it also ensures high availability, as malfunctioning components can be substituted with alternatives. They also discuss a possible means of achieving this, which suggests considerable advances can be made where others have failed.

Undoubtedly, we see many research initiatives, as suggested by the approaches detailed above. From the early inception of intelligent environments, perceptual systems have been built to observe human interactions and execute tasks. Interoperability platforms have been developed to enable seamless interoperation, while many other industry efforts have evolved to create interworking solutions. Given these approaches, we have seen end-to-end solutions for value-added audiovisual services, which are distributed via open access networks based on personalization profiles.

The plethora of devices and the communication protocols they support results in devices belonging to different domains such as the aforementioned Internet,

broadcast, and mobile. Interoperating between these domains to create a unified abstraction layer is proving to be a challenge. This has resulted in set-top box technologies that provide such abstractions, allowing devices to be interconnected and controlled. In one sense, this results in isolated islands containing devices under the control of the technology provider. Many approaches attempt to avoid this problem so as to allow a larger number of devices to form part of the Internet without forcing them into restrictive agreements.

The solution is not obvious; the problem remains of how devices and services should be dispersed, discovered, integrated, operated, and managed, thereby easing the way devices are used. This has resulted in new research disciplines that incorporate the use of semantics with advances being made through the successes of technologies such as ontology and the semantic Web. These do, however, require user involvement to create and manage workflows.

This section has presented numerous systems that attempt to address the issue of interoperability. Each approach in its own right provides benefits but also suffers from drawbacks. The challenge is to capture the positive aspects of all approaches and enable a new research strand allowing devices to be seamlessly interconnected and operated. Research conducted in isolation will not completely solve all interoperability concerns. In the following section, we describe a possible scheme to address these limitations and explain how a multidisciplinary approach may be used to incorporate the successful advances presented above. The proposed scheme illustrates how new perspectives can be applied to device interoperability, function utilization, ontology usage, and device and service management.

IV. SERVICE COMPOSITION FRAMEWORK

As we have seen, many standards exist in home networking, such as OSGi, to provide a means of abstracting device functionality as services. Coupled with the ability to create this abstraction, peer-to-peer technologies such as JXTA realize the important step of moving beyond simple file sharing, providing a vehicle to dynamically disperse and discover such services. Given the success of such approaches, it is appropriate to build on these advances. Using peer-to-peer services, we present an investigation detailing how devices can seamlessly interoperate.

The discussion describes how device operations are abstracted as services and deployed within the network, where interconnected devices are free to utilize these services. Given that all functionality is viewed as a collection of these services, compositions are automatically created based on shared characteristics. For example, a service (implemented on a device) that outputs multimedia streams may be composed with services that consume multimedia data. Given that such compositions exist, mechanisms responsible for creating relationships can manage the execution of such compositions, allowing devices to self-adapt to unforeseen changes that occur within and between services and the environment, e.g., increased network traffic. Combining these principles allows high-level applications to be automatically created through device and service interactions. This reduces the difficulties associated with combining devices, making it easier for specialists and home users alike to simply use the devices they own without having to define how applications are created beforehand.

A. Approach Overview

The design goals provide the system requirements for a suitable scheme as described in this paper. The principle goals are as follows.

- *Peer-to-Peer.* Devices and services are published and utilized using peer-to-peer technologies.
- *Abstraction.* Device functionality is abstracted as services, which can be discovered and utilized by other devices or services within the network.
- *Service Architecture.* Devices have the ability to offer zero or more framework services. If a service is not hosted by the device, then it discovers and uses the required service remotely within the network.
- *Ad hoc Services.* Devices offer and discover services without third-party intervention. Once devices are switched on, they can offer their services without having to register them with a third-party registry.
- *Semantic Descriptions.* Services are described and discovered using semantic annotations. Semantic interoperability between different ways of describing services is achieved using ontology.
- *Automatic Device and Service Composition.* Devices and services are automatically composed using semantic annotations and service capability models, i.e., devices that provide the same functionality may redundantly coexist, and consequently it may be desirable to select devices that support the best configuration.
- *Device and Service Self-Management.* Devices and services self-adapt and extend the functionality they provide beyond what they were initially designed to do. Conflicts are detected and rectified as and when they occur within the device and device configurations.

As discussed in the previous sections, there have been many proposed solutions for the interconnection of devices. While centralized approaches such as OSGi, DLNA, and task computing have generated a great deal of interest, new research initiatives such as JXTA and AMIDEN are investigating the benefits peer-to-peer technologies might offer. Initial results suggest that the

flexibility such approaches afford make peer-to-peer a viable line of enquiry where these advances can be extended to include the use of semantics. This is an approach evident in research initiatives such as CoSMoS, SWORD, and OWL-S, which has resulted in a new research community aimed at investigating how new approaches can be applied to device and service integration. In this sense, the approach presented in this paper can be seen as an important step towards networked appliance research where the line of enquiry is likely to prove fruitful. The remainder of this paper explains how the principal goals have been incorporated within the framework and highlights the novelty of our approach.

B. Device Interoperation

The proposed scheme provides a middleware that operates between devices and the application layer. Fig. 2 illustrates how all communication between compositions (Intercom, Hi-Fi, Music, and Theatre) passes through the service composition framework and the technology adapters used to communicate with devices. Not all appliances are network-enabled. Consequently, technology adapters may be required to integrate legacy devices. Nonetheless, networked-enabled appliances may choose to bypass these adapters and interact with the framework directly. This approach has already been adopted by iReady [41], where devices are interconnected using separate data communication components. Standards are provided for different network adapters dependent on the data exchanged within and across different networks. They claim their approach is flexible because device manufacturers do not have to develop, market, and sell existing products separately from network-enabled ones. Users that require network connectivity simply buy the associated adapter.

As devices connect to the network they perform four tasks.

1) They either publish the framework services they implement and/or discover and use framework services provided by other devices (the reason for this is discussed later). Devices with varied capabilities must be accommodated. In this sense, the framework needs to be flexible enough to allow devices to implement some, all, or none of the framework services. For example, a PC might implement the entire collection of framework services because it is capable of doing so, whereas a sensor might not due to its limited capabilities. In the latter instance, the sensor may discover and use framework services remotely (services provided by the PC, for example).
2) Devices publish their operational capabilities (e.g., a TV may abstract the audio and video functionality it provides as services).
3) Devices form relationships with each other and services within the network. This is not to be confused with step 1), which describes how a device implements framework services; this step relates to how devices are connected together.
4) Devices self-adapt to connections between each other and services, including environmental changes.

Many home networking standards utilize message-based approaches to dynamically discover resources. Adopting a simplified viewpoint, discovery approaches such as the simple service discovery protocol [42], as used in UPnP, provide an oversimplified means of discovering content based on keyword matching. The problems associated with this approach are well documented, resulting in the use of semantics to try and address these limitations. Consequently, the approach taken in this paper extends message-oriented techniques to include the use of semantics to describe services and device capabilities.

The rules governing semantics differ significantly over keyword matching, where logical constructs interconnect semantic elements. This does have problems of its own where heterogeneity exists between the ways concepts are defined. Given this limitation, semantic information does, however, provide a conceptual net in which a larger number of queries can be handled that would otherwise be missed. While promoting the unconstrained use of different vocabularies, there is a need to provide semantic interoperability mechanisms to help resolve conflicts. Techniques used by the semantic Web [8] and semantic Web services communities can help to address this issue. We can therefore use unconstrained vocabularies in line with semantic interoperability mechanisms to help interconnect devices and services. In achieving this, the approach adopted by Paouluci *et al.* is extended to not only discover services but also help automatically compose services with little or no human intervention.

Building on peer-to-peer technologies, home network solutions are investigating how devices can use the techniques they afford to automatically integrate themselves within environments devoid of any third-party

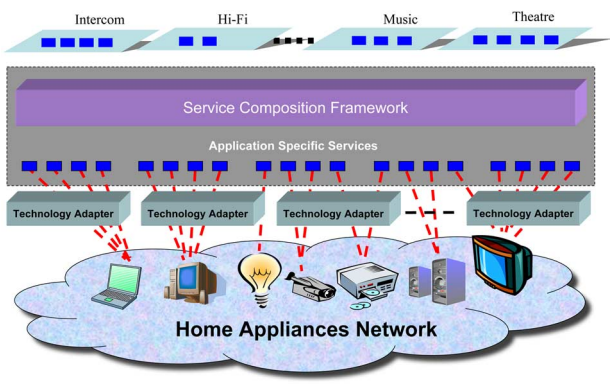


Fig. 2. ***Proposed framework.***

intervention. Communication protocols such as Bluetooth and Zigbee are now demonstrating how personal area networking can be achieved, allowing devices to join networks without having to register themselves or the services they provide. Perhaps the most popular of ad hoc networking within the home is that of UPnP and AMIDEN [20].

The extension of such approaches to incorporate the use of semantics and self-adaptation enriches them to better enable devices to automatically interoperate. Using semantically annotated service advertisements, devices can advertise the functionality they provide by publishing them within peer-to-peer networks. Depending on the technology, advertisements are likely to have a time-to-live value that expires after a given time period—consequently, advertisements have to be periodically republished to keep the service alive. Devices may fail unexpectedly. In this instance, service advertisements would no longer be republished and would eventually be purged from the network. Early results suggest that our approach, based on these principles, helps improve device and service deployment as well as provide additional benefits that allow devices to self-adapt.

Using a primary service [43], as illustrated in Fig. 3, it becomes possible to map devices onto any middleware standard such as JXTA, RUNES, OSGi, or UPnP. Devices can join the network as specialized or as simple networked appliances. Specialized networked appliances may have the ability to host services, store and evolve semantic information used to describe and discover services, and propagate service requests within the network. A simple networked appliance by definition may not have these capabilities. This type of device would typically join the network, propagate queries, and invoke discovered services, or simply just be controlled. An example of such devices might be a sensor or an X-10-enabled lamp

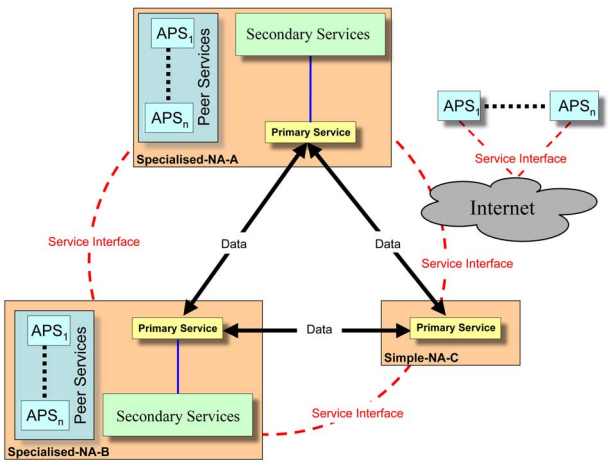


Fig. 3. ***Connecting capable and less capable devices.***

[44]. This would enable any device, irrespective of its capabilities, to effectively choose how it wants to interact within the network.

Fig. 3 illustrates two extremes—networked appliances that are highly capable [*specialized networked appliance* (NA)] and those that have limited capabilities *(simple NA)*. However, it is envisaged that there will be a myriad of other possibilities between these extremes.

Given the configuration in Fig. 3, the primary service is the only compulsory service each device needs to implement. This service is responsible for marshalling all communications between framework services (secondary services) and *application peer services* (APS), whether they reside locally or remotely within the network. If *simple-NA-C*, as illustrated in Fig. 3, is a sensor (with very limited capabilities), it could use the secondary services provided by *specialized-NA-A* (e.g., a personal computer). The primary service on the sensor marshals communications to the hardware of the sensor and the primary service located on *specialized-NA-A*. For example, the sensor could be responsible for reading data and providing it to *specialized-NA-A*, where aggregation could take place and results periodically returned to the sensor or some other persistent storage service provided by another device, e.g., *specialized-NA-B*. The secondary services themselves provide framework functionality, which could include aggregation, while application peer services provide abstractions to the operational functions devices provide, such as biofeedback readings provided by heart and galvanic skin response sensors.

When devices initially join the network, the goal is to try and discover devices they may have relationships with. For example, a treadmill machine in a gymnasium may discover all the equipment used by someone during a training session. This configuration may also include a medical interface used to submit information to the user's general healthcare practitioner, whereas their mobile phone could be used to synchronize with healthcare services on the users home PC designed to monitor fitness levels.

By managing references to external services (relationships), devices can adapt to changes that occur. For example, if the user's mobile phone becomes unavailable for some reason, the treadmill could detect this change and automatically select an alternative device, such as an Internet information processing service provided by the gymnasium. Looking at another example, depending on noise levels within the gymnasium, the user's MP3 player volume could be increased or decreased to accommodate this change. It could even mute the music if a telephone call is received. In this way the collective operations of all devices form high-level applications, where each device monitors its own interaction within the composition and adapts to any changes that may occur.

One of the key requirements, and one supported by the RUNES project, is that every device must have the ability

to form part of the larger network. While this provides obvious benefits—such as the ability for similar functionality to exist simultaneously on multiple devices within the network—it also raises questions about the quality of compositions created. In particular, care must be taken during composition to ensure that the most appropriate functionality is chosen amongst ostensibly similar devices or services. Devices and services must be composed based on the best resources available. In this sense "best" could mean resources that provide the best functionality, or resources that are free to use. Whatever the definition, the term needs to be identified and adhered to. In this paper, it refers to the service and the ability of the device to provide that service.

As a result, it is beneficial for devices to describe their capabilities in terms of how well they execute the services they provide. This means that devices would benefit from providing a corresponding device capability model (this is discussed in more detail in the following section) describing how capable it is. This allows devices to process capability models to determine how well the device supports the previous situation before including it in a composition. For example, a mobile phone may be connected to the gymnasium's satellite system, allowing a user to listen to a music channel. If the user moves to a different location, a plasma display may be selected because it provides a better entertainment experience (audio and video). If the user changes location again (to alternative gymnasium equipment) where the plasma display is no longer visible, the configuration may adapt to redirect the satellite signal back to the mobile phone.

Describing device capabilities, coupled with semantic descriptions of services, underpins the approach taken in this paper to seamlessly interconnect and self-adapt networked appliances.

V. SECONDARY FRAMEWORK SERVICES

To reiterate Fijii's argument [33], despite the difficulties associated with dynamic service composition it provides several benefits, notably flexibility, adaptability, and availability. While Fijii talks about dynamic service composition, the argument is generally applicable to service-oriented architectures. Distributing functionality as services, including framework and application-specific services as discussed above, ensures high availability. The subtle advantage is that it becomes irrelevant, in part, who provides these services or where they reside. In keeping with the vision of RUNES, this becomes particularly important in networked appliance research where devices can form part of the larger network irrespective of their capabilities. Some devices may not have the ability to host some or all of the required services, and for this reason the proposed scheme combines service-oriented computing and peer-to-peer technologies to disperse and enable framework services. More capable devices host these services, and devices with fewer capabilities simply discover and use them remotely. This enables the inclusion of a larger number of devices, which are free to decide how they join the network, what they will offer, and what they require.

We turn our attention now to describing how the secondary services extend the primary service functionality discussed in the previous section to allow devices to dynamically compose as illustrated in Fig. 3. The process requires the automatic resolution of terminology differences between vocabularies, as well as assessment of device capabilities. Importantly, the framework is not limited to these services. It has the ability to accommodate any new functionality that may be required as the framework evolves.

A. Dynamically Composing Devices

As we have seen there are numerous ways of interconnecting devices. DLNA uses open standards coupled with proprietary manufacturing, while task computing uses semiautomatic service composition techniques where the user combines services to perform some task. While adaptations of such approaches may evolve, a point will be reached where simply deploying networked appliances and services will not be acceptable because it will be too difficult for users to combine them. We therefore envisage that approaches more closely resembling projects such as RUNES will prove most successful.

To address the issue of increasing complexity, it is important to ensure that management tasks are considered in parallel with networked appliance development. We believe the management of devices and services to be perhaps one of the most important aspects of networked appliance research. For this reason, devices must be equipped with management techniques to automatically integrate and adapt services used in high-level applications. For example, when a networked-enabled media player is connected to a home network, it must automatically advertise its services and form relationships with visual and audio devices, such as TVs and speakers connected to the network. Underpinning this dispersed operational functionality, management services need to react to adverse environmental conditions and composite conflicts, such as the unavailability of services [24].

Many approaches discussed in this paper rely on the user to compose and then manage compositions once they have been executed. Moving beyond this requires devices to codify this process and self-manage the internal operations they support and adapt to adverse effects. Approaches such as task computing, CoSMoS, and SWORD argue that utilizing semantics may form a basis to achieve this. Following this interesting line of enquiry, we extend our investigation to the automatic combining and self-management of devices.

Service ontologies allow services to be semantically annotated, which can be embedded within service

advertisements. The same ontologies can also be used to describe service requests. There are currently four main service ontologies. The *service ontology*, *service profile*, *service process model*, and *service grounding* [45]. Service ontologies allow services to be described at an abstract level in terms of *inputs*, *outputs*, *preconditions*, and *effects* (IOPEs). The IOPEs form explicit relationships between the different ontologies, which are in turn mapped onto the underlying service interface [for example, Web service description language (WSDL)]. In this way, the service ontologies and the service interface provide a mechanism to link semantic descriptions to the implementation details used to describe how services are invoked. This process helps independent services offered by devices to be discovered, composed, and executed with little or no human intervention.

Using these principles, functions (the operational capabilities devices support, e.g., a heart-monitoring sensor provides functions to read and transmit heart-rate information) can be discovered and composed to create high-level tasks or applications (service compositions). Looking at high-level functionality Fig. 2 shows how a virtual intercom system (a device that does not physically exist but rather emerges through the interaction between different services) might be created by combining the microphone from a mobile phone and all the speakers provided by appliances throughout the home, such as the TV, surround speaker system, and PC. Virtual appliances, like our intercom system, remove the need to physically install custom solutions. This differs from proprietary solutions such as iReady and, more importantly, current home appliances, which often do not support networking capabilities.

Using the concept of IOPEs, devices and services are composed by matching similarities between the service request and the service ontologies. Service ontologies, in conjunction with the domain ontology, are matched if vocabularies are syntactically different. An example is illustrated in Fig. 4.

The DVD player submits a service request to the digital camcorder (step 1). Upon receiving the request, the composition service on the digital camcorder begins by iterating through the *service profiles* for each service it provides (step 2). Using the domain ontology, the IOPEs in the service request are matched with IOPEs in the service profile. If exact matches are found, then the process simply moves onto the next IOPE. However, if syntactic differences are found, the two terms are passed to the domain ontology [46] (steps 3 and 4). If a relationship exists between the two terms, a match has been found that semantically links the two terms together. For example, in the figure, the domain ontology finds a relation between the term *Movie* and the term *Film* (step 5). While they are syntactically distinct, the domain ontology shows that the term *Movie* is linked to the term *Film* via an *equivalentTo* relationship.

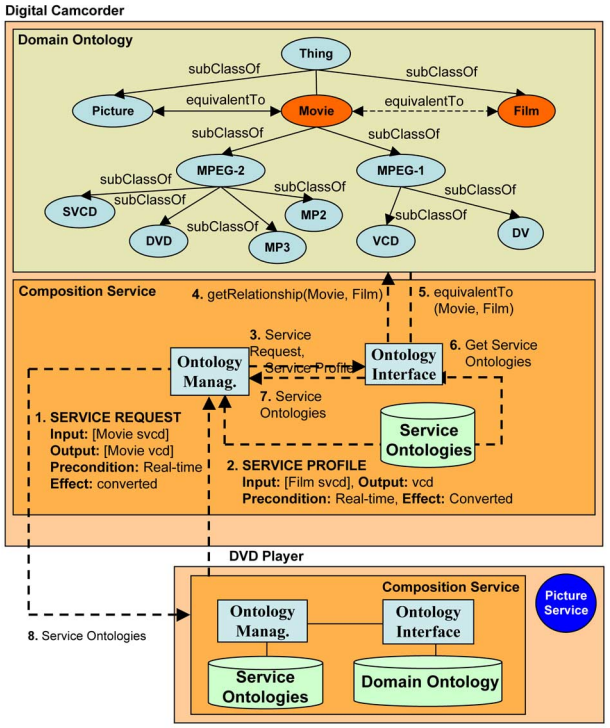


Fig. 4. ***IOPE matching.***

During the matching process, a table is created containing the matched IOPEs from the service request. The matched IOPEs act as keys in the table and have corresponding values, which represent the names of the IOPEs used in the service ontologies. This process is important because the service request and service ontologies may refer to semantically equivalent IOPEs differently—the table of key-value pairs (stored on every device capable of performing semantic interoperability) creates a semantic mapping between the different terms used.

If all IOPEs in the service request are matched, this constitutes an abstract match. This results in the device's retrieving all the service ontologies relating to the service profile, along with the service interface (step 6). The ontologies along with the interface are then returned to the DVD player (steps 7 and 8).

To determine whether the IOPEs defined in the service request can be bound directly onto a signature (a method with supporting input parameters and a return value) defined in the service interface, the service profile is processed and the values associated with each IOPE are retrieved. These values specify which *atomic process* each IOPE belongs too in the *service process model*. The IOPEs may have been matched at an abstract level; however, they could belong to different atomic processes. Therefore, the framework tries to determine if a single atomic process exists that supports all of the IOPEs in the service request. If so, the operation name can be extracted from

the *service grounding* along with the parameters, parameter order, and endpoint address from the service interface. Signatures can then be created from this information and used to bind to and execute the required service. This is an important requirement that extends the approach used by Paolucci *et al.*, who only consider the *service profile* to discover services so that the user is required to use the remaining services ontologies to manually form compositions between services found.

It is well documented that automatically composing devices is problematic. This can be attributed to the variation in how service interfaces are defined and described, where one single difference of parameters in the signature can render the service inappropriate for the composition. To accommodate this, mechanisms could be adopted similar to constructors used in object-oriented programming, where a base constructor could be used and then extended to include the different ways the object can be created. While this is one possible solution, it would be difficult to predefine every possibility. A more effective way to address this limitation is to extend the concept of automatic service composition to enable signatures to be composed, resulting in new signatures' emerging. We achieve this using intermediary services and extended interfaces.

Using this approach, devices can form compositions between devices or services either directly or indirectly through signature rewriting. For example, in Fig. 5, "DVD Player 1" reads the data from an inserted movie disk. The "Player" service discovers that the media format is Xvid, which it cannot process because it only has an MPEG-2 decoder. DVD Player 1 tries to resolve this conflict using an intermediary service (a service provided by another device), which takes as input an Xvid data stream and generates an MPEG-2 output stream.

Reformulated service requests can be propagated in the network describing the IOPE requirements as Fig. 5 shows. DVD Player 1 finds DVD Player 2, which can indirectly stream an Xvid movie into an MPEG-2 media format using a service provided by a laptop. DVD Player 2 can use the laptop to convert the Xvid format into a DivX format, which it can process and convert into an MPEG-2 stream. DVD Player 1 may not be aware of the composition between DVD Player 2 and the laptop—it is only concerned that DVD Player 2 can successfully convert the Xvid data into MPEG-2.

This process can be achieved using an extended interface metadata object. It describes how signatures are constructed and indicates whether the intermediary service itself can be directly invoked or also requires intermediary services. This process allows services to dynamically discover and resolve IO conflicts that may occur and proactively establish compositions with intermediary services. This may result in several candidate services that provide the same functionality. Services that best match the device capability requirements defined in the service request are added to the extended interface metadata object.

The extended interface (EI) service, illustrated in Fig. 5, is invoked when a service provided by the device does not directly support a method invocation. This service has a fixed operation name called EI that takes two parameters—the first is the extended interface metadata object and the second is an array containing all the parameters required to invoke the service. This service processes the extended interface metadata object, which provides information about the operation name for the intermediary service, the parameters it takes, including the associated data type information, and the order in which the parameters appear in the signature. In effect, this explains how a method invocation should be constructed, i.e., as the elements in a Web service description language (WSDL) document do when binding to and invoking a Web service.

If the connection mode is "direct," the extended interface service uses the metadata for the intermediary service to construct the required signature using the parameters in the array, before binding with it and executing the required method. In this instance, "direct" means that device A can directly use a service S_1 provided by device B without having to use any intermediary services. If the connection mode is "composite," the extended interface service processes the extended interface metadata object for the intermediary service it needs before connecting to its extended interface service and

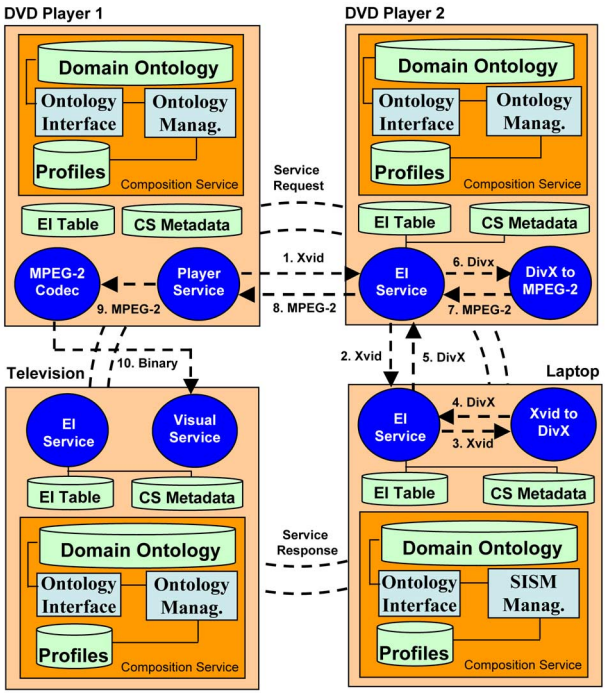


Fig. 5. ***Dynamic service composition.***

passing it the metadata and the parameters. In this instance, "composite" means that device *A* indirectly uses a service S_1 provided by device *B* via service S_2 provided by device *C*. This process continues until a "direct" connection with a service in the composition is established.

Using these techniques, service interfaces can be evolved over time to accommodate inputs they were not initially designed to process. For example, a DVD player that only implements an MPEG-2 decoder can read a number of different media formats and interact with the "visual" service using data adaptation services as described above.

B. Managing Distributed Device Ontologies

The crucial aspect that allows networked appliances to be automatically composed is the use of semantic annotations to describe services and domain knowledge. As we have seen, networked appliances provide zero or more services. For example, a television offers visual and audio functions, as well as terrestrial communication channels. Discovering and utilizing these functions within and across different networks with current approaches presents a significant problem. Although initiatives such as task computing, SWORD, and CoSMoS try to address this using semantics, the main interpretation of the semantics in the context of the composition can only be achieved by the user. Given the problems, many approaches such as UPnP and OSGi do not even consider semantics, opting instead to use simple syntactic descriptions. Disassociating themselves further, approaches such as ePerSpace, DNLA, and iReady decide on more proprietary solutions, clearly defining how devices are described and interoperated. Alternative approaches are clearly required. As such, we build on the concept of ontology, as defined by Uschold *et al.* [27], and argue that ontologies may help solve a number of limitations inherent in other approaches.

Typically, ontologies have been managed by centralized consortia who agree what information an ontology should contain and how it should be described. This is subjective and difficult to maintain. An alternative, more flexible approach is to allow devices in a peer-to-peer network to collaborate, allowing ontologies to be automatically evolved. This would facilitate devices by extending and/or reifying concepts they already have. Furthermore, it would allow devices to understand different vocabularies used by other devices. Coupled with peer-to-peer technologies and the principles of general consensus (what the majority believe to be true), this would move away from centralized ontology management and result in a more scalable approach. Each device is treated as a self-contained knowledge node, which through interactions and self-management algorithms allow global ontologies to emerge [46].

Adopting this strategy, we have followed a path of enquiry that suggests this approach could prove fruitful. When devices join the network, they initially have knowledge to support the vocabularies they use to describe the services they provide. Over time, device ontologies are extended to include those used by other devices. By propagating queries, concepts can be evolved by utilizing those provided by other devices. Devices process requests, which are used to query the device's local domain ontology to determine if any concepts exist relating to those defined in the query. Any concepts found can be extracted along with any dependent concepts and associated relationships and returned to the querying device. This may result in several responses, containing subjective ontological structures that lead to structural and possibly lexical variations between all responses received. To accommodate this, structural patterns (commonalities found within all ontological structures) can be extracted and used to produce an optimal structure that captures the general consensus.

To demonstrate this approach, Fig. 6 illustrates a subset of a device's ontology C1 and three responses R1–R3. This represents the results the device has received from the network after a query has been propagated. It is clear that although structurally R1–R3 are different, commonalities exist within all structures. For example, given the nodes Movie and MPEG-2, the framework can determine that these nodes are common by calculating the number of times they occur in all structures. In this instance, both nodes appear in all four structures. Therefore, they will appear in the optimal structure. In contrast, the nodes Film, VCD, DVD, and MP3 are low-scoring nodes because each node appears in one structure only. These are classed as low-scoring nodes, and the probability of them appearing in the optimal structure is reduced.

Using two fitness functions [46], nodes and relationships can be extracted and used to build the optimal structure. The higher the fitness function, the more specific the extraction process is. In contrast, the lower the fitness function, the more general the optimal structure will be. Setting fitness functions is dependent on a number

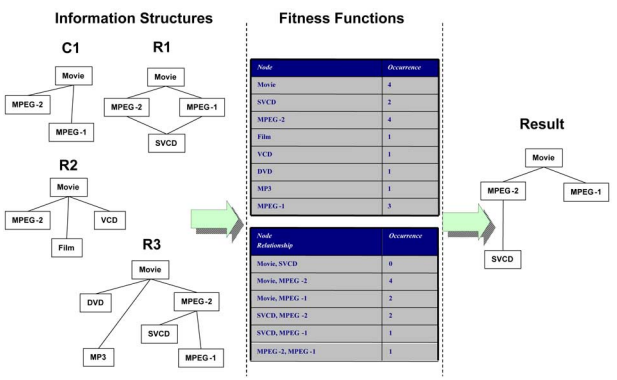


Fig. 6. ***Statistical pattern extraction.***

of factors, such as accuracy, concept, size, and processing time. For example, a mobile phone, which is processor and memory restrictive, will set its fitness functions high to ensure that less common information is excluded. This is assuming that a mobile device would explicitly implement an ontology processing service at all (devices with limited capabilities may choose to use such services remotely).

Returning to our example, the first fitness function places all the unique nodes found in the device's current structure C1, and in the responses received R1–R3, into an array. Each node is given a fitness value based on the number of times it appears within all structures. This is called *term frequency*. For example, the node Movie is given a fitness value of four because it appears in all structures. The second fitness function places all the unique relationships that exist between any two nodes and places them into an array. Again, the fitness value for each relationship is calculated based on the number of times a relationship occurs in all structures. In this paper, only two fitness functions have been used; however, the ontology service could accommodate any number of fitness functions to improve the evolution of ontologies.

Once a list of ranked nodes and relationships has been created, they can be used to produce a binary vector. Each position in the vector represents a node or relationship. A value of one in the structure is placed in the vector based on the presence of a node or relationship in that structure. Optimal structures can be obtained by specifying that the optimal binary vector must have the top n nodes and the top n relationships. For example, if the nodes and relationships in C1 are used as illustrated in Fig. 6, the following vector, ordered based on fitness, could be produced:

$$[1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0].$$

The first eight bits represent the nodes and the last nine bits represent the relationships. The position of the bit is important because it corresponds to a particular word representation. For example, a word version of the above vector may be as follows:

[Movie, MPEG-2, MPEG-1, SVCD, Film, VCD
DVD, MP3, Movie|MPEG-2, Movie|MPEG-1
Movie|VCD, Movie|Film, Movie|DVD
Movie|MP3, MPEG-2|SVCD, MPEG-1|SVCD
MPEG-1|MPEG-2].

If a value of one is in the third position, then this means that the structure has a node called MPEG-1. Furthermore, if a value of one is in the fifteenth position, this means the structure has a relationship between the node MPEG-2 and SVCD. For illustrative purposes, the *Result* structure in Fig. 6 can be generated by using the top four nodes within the vector of nodes found and using the top three relationships. This means that the optimal structure, given these rules, would be as follows:

$$[1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0].$$

Optimal structures can then be used to evaluate each individual structure to determine how fit each structure is based on a given metric. For example, each matching bit in the vector might cause the structure's fitness value to be incremented by one. With the optimal vector illustrated above, we can say that an optimal structure has been obtained when a given structure has a fitness value of 17.

There are several ways of obtaining the optimal structure. Simple statistics or feedback loops could be used to evaluate how successful compositions are. The approach presented in this paper provides functions for statistical analysis and genetic evolution. However, in the latter case, the user is required to input the optimal structure required. Future work is needed to evaluate the effectiveness of such an approach.

Optimal structures can be merged with the device's local ontology by extracting the nodes from the optimal structure and checking whether they exist in the device's local ontology—any missing nodes are added. Once this is complete, the process can be repeated for all of the relationships that exist between nodes in the structure. These principles allow ontological structures to be managed within and across all devices connected within the network, where the collective knowledge provided by all devices is seen as a distributed ontology.

General consensus provides the basis for this approach and is similar to that provided by Stephens *et al.* [47], which we have extended to manage and evolve semantic serializations used to describe device functionality. This can overcome the difficulties associated with centralized ontology management approaches such as PROMPT and ONION [48], [49], where subjective opinions are replaced by a consensus algorithm that emphasize popular concepts, while deemphasizing those that are less popular. Deemphasized concepts are eventually removed from the ontology. This provides a continually evolving semantic interoperability mechanism allowing devices that use different terminologies to talk to each other nonetheless. Ontological concepts are evolved based on commonalities between all concepts relating to the structure to be evolved, within the network, and the functionality the device provides. This restricts the management of concepts by devices to concepts they are only interested in. For example, a mobile phone will not contain or participate in the evolution of ontological structures other than those relating to the functions it provides. Therefore, only concepts that are relevant to

the description and interactions performed by the device need to be contained and managed. Early results suggest there are significant advantages over existing ontology evolution approaches [46].

C. Enhancing the User Experience

To fully utilize devices, we need to determine what functionality is available and how effectively devices providing those functions can execute them. For example, a plasma television and a video-enabled mobile phone are both able to process video information. However, the quality of these services will differ. A plasma television may be considered better equipped to process high definition video compared to a mobile phone. Given these different capabilities, selecting and composing devices must be dependent on quality of service and application-specific requirements. To address this, devices have the ability to select devices and services based on how well they can execute a service [23]. This can be achieved using metadata models, dynamically generated to assess device capabilities. Each model contains information about every resource the device manufacturer deems important. Using this information will allow a device to rank functionality based on ordered capabilities where the top of the list provides highly capable devices and the bottom those that are less capable.

Several steps can be performed to assess a device's capabilities. First, the device's current computational load and the additional load required to execute the desired service can be calculated. By adapting existing approaches [50], [51], the formula defined in (1) can be used to calculate the quality of a required resource, where a resource r offers a service s that requires $ac_{s,r}$ units of some total resource value tr_r

$$resc_{s,r} = \frac{ac_{s,r}}{tr_r}. \tag{1}$$

This formula can determine what proportion of some resource will be used given the total value of the resource available. For example, if a service requires 1 Mbyte of RAM and the device provides a total of 32 Mbyte, then the service requires 3.1% of that total.

In parallel, devices can determine if executing the service will overload its computational capabilities. For example, if the device has 75% CPU utilization, the device may struggle to take on the increased computational overhead if a required service is executed. Determining what constitutes a device's being overloaded is application specific. For example, if the service is a transcoding service, the application may state that CPU utilization should be no more than 10% because transcoding is processor intensive. Conversely, a service that processes simple commands such as "light switch on" may require nominal computation; thus a device that has 75% CPU utilization could accommodate the additional computational overhead without any adverse effects.

Quality of service may also be affected through shared computation. By calculating the overhead for each resource the service request deems important, a comparison can be made with the device capability model defined in the service request. This can be achieved using a technique called multiattribute utility theory (MAUT) [50], [51], which produces an overall capability score for some device D given the attributes defined in the device's capability model. This formula can be defined as

$$DCScore(D, DCM) = \sum_{i=1}^{d} cw_i(DCM) \cdot D(v_i), \tag{2}$$

where $DCScore$ is the overall capability score for device D according to the device capability model DCM, d is the number of capabilities for the type of device, $cw_i(DCM)$ is the importance rating of attribute i according to the device's DCM, and $D(v_i)$ is the status rating for attribute i. The importance rating describes how important a given attribute is in relation to all the attributes used, e.g., the CPU attribute may be the second most important attribute with an importance rating of 30, meaning that the CPU is considered three times more important than an attribute with an importance rating of ten. The status rating describes how well the device supports a particular attribute, e.g., a device may have "excellent" for its CPU attribute, which may equate to a value of 75.

Given the two formulas, the device's operating system can calculate the service ratings programmatically by estimating the average attribute values and assigning the appropriate status ratings. For example, if the device uses on average 25% of its CPU when the required service is executed, the operating system may assign the CPU_Load a status assessment of "excellent" with a status rating of 75.

Equation (3) can amend the MAUT formula to take into account the current resource load and the load required to execute a service. In this instance, $DCScore$ and $resc_{s,r}$ are multiplied to give a combined resource load value, indicating whether the device can effectively execute a service it provides

$$DCScore(D, DCM) = \sum_{i=1}^{d} cw_i(DCM) \cdot D(v_i) \cdot (1 - resc_{s,r}). \tag{3}$$

When (3) is used to calculate the score for the device capability model, it is compared with the score generated for the device capability profile. If the device capability model score is equal to or greater than the score for the

device capability profile, the device is said to be capable of effectively executing the service while ensuring the quality of service is maintained. In this instance, the service details can be returned to the service requester. Using this technique provides a computationally inexpensive mechanism that allows devices to proactively select the best devices and functionality that increase the overall quality of device and service compositions.

This section has considered all of the secondary services that compose our proposed framework. The following section demonstrates their practical use in an intelligent home environment.

VI. INTELLIGENT HOME ENVIRONMENT

In this section, we discuss how the services can be combined to create a distributed service-oriented platform for use within an intelligent home environment. In this way, the implementation can be used to:

i) test the proposed design decisions and illustrate how the collective functionality described addresses the limitations with current approaches;
ii) demonstrate how the operational capacities provided by devices can be dispersed and better utilized;
iii) underpin interconnected devices with management services for self-adaptation.

The demonstration prototype includes two wirelessly connected televisions: a wireless media player and two wireless audio speaker systems. The televisions host video and audio services able to process video and audio data streams. The media player hosts a player service for outputting MPEG-1 multimedia data, and the audio speaker systems host audio services able to process audio data streams. The prototype forms part of a bigger research strategy, which has also been successfully implemented on sensor networks (crossbow, ember, and TINI microcontrollers) to control home networked devices using biofeedback [52], [53].

When devices are initially switched on, they perform the four tasks outlined earlier in Section IV-B. That is, they first publish the services they provide. Secondly, they publish the application specific services providing access to the device's operational capabilities. For example, the television publishes its audio and video services, the media player publishes its player service, and the audio devices publish their audio services. Thirdly, they try to discover other devices within the environment they have a relationship with. For example, the media player tries to discover devices capable of processing audio and video streams outputted by the player. Fourthly, they monitor the communication links they have with other devices and self-adapt to changes that occur. Using the simple control interface illustrated in Fig. 7, users can discover, use, and control any device in the network, including the services it provides.

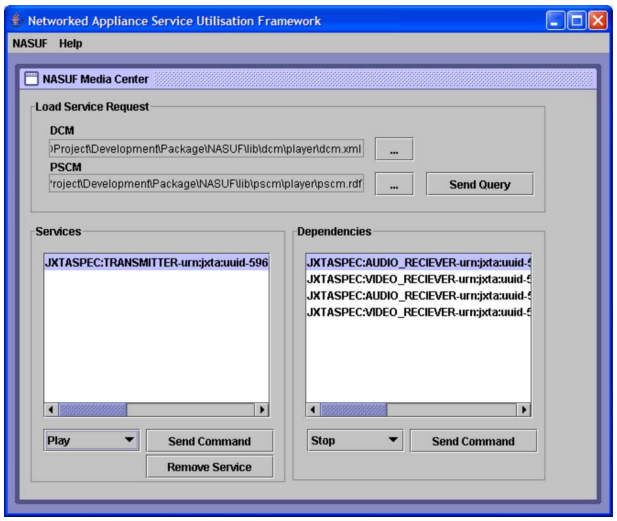


Fig. 7. ***Device user interface.***

Users select the device and service capability models to describe the quality of service factors the device must support (serializations were created by making Java Native Language calls to a C dynamic link library capable of querying the device's operating system) and the service functionality required. This is achieved using the interface illustrated in Fig. 7. Models are serialized as XML and appended to service requests before being propagated within the network using the *Send Query* button. Discovered services are added to the *Services* list box. Any dependency services are added to the *Dependencies* list box. For example, the player service discovered in Fig. 7 has four dependency services: two audio and two video (these are automatically discovered when the device is initialized—there could have been more, but this was all that could be found at the time). While dependency services are automatically discovered by the player without any human intervention, controlling specific devices will require the user to operate specific functions. For example, the user needs to press the play button on the networked-enabled player before the multimedia streams are sent to dependency services. What is important is that the user is not required to create the composition; this is achieved automatically by the framework services.

The dropdown box directly under the Services list box contains all the commands that can be sent to the player service. These are extracted via the service ontologies, from the service interface (WSDL file). Any of these commands can be selected and sent to the device using the *Send Command* button. For example, selecting *play* and pressing the *Send Command* button results in the player streaming *audio* and *video* data to the audio and video dependency services.

The prototype demonstrates how devices self-adapt when exceptions occur within compositions. When the

user sends a *play* command, the player instructs the audio speaker system and the television to begin processing the media streams sent from the player. For demonstration purposes, the audio speaker system was removed from the network while a movie was being played. This conflict is detected by the media player, and a previously discovered audio service is instructed to begin processing the audio data. The disconnected service was then reconnected to the network. The media player sensed this change and compared the device capability model for this speaker system with the device capability model for the current speaker system being used. It discovered that the newly published speaker system provides a better auditory experience than the speaker system currently in use. In this instance, the media player instructs the current audio speaker system to stop processing the audio stream and instructs the newly published audio speaker system to begin processing the audio stream instead. This functionality can also be used to self-adapt the video services.

Rather than using a centralized monitoring service, devices are empowered to monitor the local connections they have between other devices and services they use. This allows management to emerge through low-level changes making our approach highly scalable. This is on one level similar to the reactive system proposed by Chakraborty *et al.* [40], contrasting clearly with approaches such as OSGi , task computing [6], [7], and SWORD [31].

A. Technical Details

In developing the prototype, a multidisciplinary approach has been adopted whereby several tools and standards have been combined in the implementation. This is not simply an integration exercise but rather a novel approach that provides a new perspective on seamlessly interconnecting devices. The primary and secondary services have been developed using the JXTA protocols. JXTA allows any device to be connected to the network independent of platform, programming language, or transport protocol. This includes simple networked appliances such as those based on X-10.

JXTA wrappers can be added to X-10 nodes in the PLC network, thus exposing them as services where the pipe advertisement ID corresponds to the concatenation of the *houseCode* (letter between A and P used to define groups, for example, each room in the house could be grouped using *houseCodes*) and a *deviceCode* (number between 1 and 16 used to address devices within the group (i.e., C5). Any X-10 device on the PLC network can be controlled by discovering the corresponding JXTA service, using the *houseCode* and the *deviceCode* (C5), binding to it and sending commands, i.e., on, off, dim, bright) using Peterson's X-10 API.

Framework services are predetermined, and each device understands how to discover and invoke them. Bindings between devices and secondary services are achieved using pipe advertisements. Pipes are used by services and applications to send and receive information between the two. Through the abstractions provided by JXTA, they can be viewed as virtual communication channels, much like sockets programming, except pipes exist independent of location and network topology.

Secondary services providing the same functionality use the same pipe advertisement. This ensures that devices do not continually create and publish new advertisements each time the device is connected to the network. This minimizes discovery overheads and ensures that advertisement caches do not inflate over time. All secondary services have been developed in Java and implemented within the service layer of JXTA. This is a necessary step that extends JXTA to manage how services are dispersed.

Application-specific services are used to expose the operational capabilities of all devices. Many functions may exist, consequently pipe advertisements used are not known by devices beforehand. As discussed in Section V-A, service ontologies describe device functionality by mapping high-level semantics onto low-level service interfaces. This is achieved using OWL-S [45]. The OWL-S standard is also used to describe service requests (serialized as *Service Profiles*), extending the service discovery mechanisms in JXTA to incorporate semantics. This is similar to Edutella [54], [55], where semantic discovery is incorporated into peer-to-peer networks. However, they focus on discovery alone and do not consider automatic service composition or networked appliances. Using matching algorithms we have developed, IOPEs in the service request are matched with IOPEs in the service ontologies. Matches result in service ontologies' being returned to the querying device. This provides an effective mechanism to discover required services. It improves the approach taken by Poalucci *et al.* [29], [30] who only use the *Service Profile* to find services, where we use the remaining service ontologies (*Service Process Model* and the *Service Grounding*) to map IOPEs onto signatures defined in the service interface.

The framework is underpinned with semantic annotations to address ambiguities between the different vocabularies device manufacturers use. Mechanisms have been developed to use ontologies to help address semantic interoperability. Resolving ambiguities between terms that are syntactically distinct but semantically related is achieved using a distributed domain ontology serialized as OWL-DL (a constrained ontology serialization that can be used by most reasoners). This is called the networked appliance ontology, and the first level is illustrated in Fig. 8 below.

The ontology itself currently has about 500 concepts that semantically describe common household appliances and their associated properties such as inputs, outputs, and events. Individual ontologies for different household appliances were developed using the Protégé 3.1 ontology

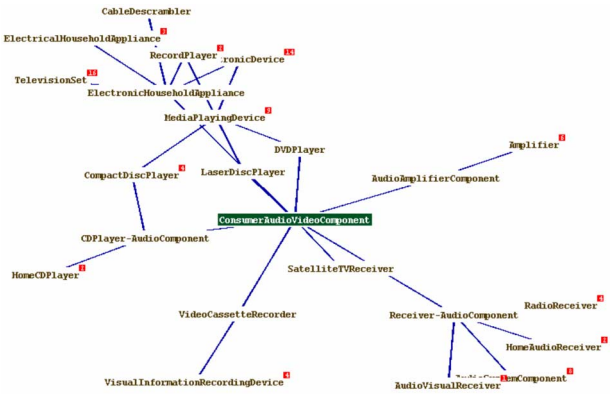


Fig. 8. ***Networked appliance ontology.***

editor and the OWL plug-in [56]. Ontologies are processed using the Protégé-OWL API.[6] Once deployed, they are continually evolved over time. The prototype demonstrates that this helps devices learn the vocabularies used by different devices.

The Protégé-OWL Reasoner API[7] is used with the Racer reasoner [57] to process these serializations. Jena [58] reasoners were considered. However, a number of performance problems were encountered. For example, when an inferred model was created using internal and external reasoners, out of memory errors occurred. Nevertheless, Jena is used to perform simple querying on OWL-S serializations. This is achieved using the ontology models provided by Jena and RDQL [58]. In some instances, it is easier to query Resource Description Framework (RDF) serializations using RDQL rather than using an inference engine, which can be computationally more expensive. Furthermore OWL-S serializations do not conform to description logics, so it is difficult to load them into reasoners such as Racer.

Audio and video application services were developed using Java. Media streams were broadcast using the Java media framework and the real-time transport protocol.[8] We developed a conversion service to transcode AVI encodings into MPEG-1. The JFFMPEG API[9] was used to perform this transcoding.

Devices and services are managed using the *Device* abstract class. Every device instance subclasses this class. The Device class provides management functions to marshal the relationships between devices. This is a custom management component we have developed in Java that uses caching techniques in conjunction with threaded listeners to monitor incoming messages. Management services have been developed to continually monitor and change compositions to extend and maintain quality of service requirements. This functionality extends many current home networking and networked appliance initiatives. The proposed approach enables zero-configuration and allows devices to self-manage. Using the tools defined in this section, we successfully developed the working prototype that implements the intelligent home environment described above.

VII. FUTURE WORK

Research into networked appliances is developing rapidly, and we have looked at some of the fascinating projects and technologies that exist in the area. We believe our own work is testament to the exciting possibilities that networked appliances engender and will provide a foundation for further work in the area of networked appliance management in home networks. A number of issues remain to be addressed in the area, for example, the ontology evolution mechanisms are problematic—only simple serializations can be processed. Nevertheless, our approach can apply new perspectives to distributed ontology evolution and semantic interoperability. It illustrates how simple taxonomical structures can be constructed and evolved over time to enable basic semantic interoperability. We believe further research is required but will allow such issues to be resolved in the future.

Security is an issue, and ad hoc environments raise an important question regarding the trustworthiness of service providers. Although we have started to address this issue [59], it falls outside the central theme of this paper. Service composition mechanisms require further research to include better matching algorithms. Feature interaction [60] is also challenging: services may operate well when used in isolation or within small compositions but cause problems when a large number of services or devices are used at the same time.

Furthermore, user preferences have only been weakly addressed in this paper and related work, and the same can be said for contextual issues. It is important that these requirements are addressed as simply restricting users to the collective formation of relationships of devices alone may be insufficient. In our example, we use a mobile phone and a plasma TV, but it may be that the user is about to go mobile, so the mobile phone would be the preferred device. This would obviously be dependent on the context of the user. Without such understanding, we will fail to see a solution that satisfies the competing but important needs of networked appliance management in home networks. Initial work on this has already begun [23], [52], but we could also benefit from research efforts such as task computing, SWORD, and CoSMoS in addressing these issues.

There are also scalability issues relating to the use of peer-to-peer technologies—in a partitioned world where a relatively small number of devices may be used n^2 versus

[6]http://protege.stanford.edu/plugins/owl/api/index.html.
[7]http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html.
[8]http://java.sun.com/products/java-media/jmf/.
[9]http://jffmpeg.sourceforge.net/index.html.

2∗n issues may not be an issue. However, problems do arise when you start incorporating WANs as such further work is required.

The environments and devices discussed in this paper are difficult to debug, trace, and audit because of their distributed nature. Further work is required to develop APIs and standards in order to establish a *world view* to better understand the current status of devices. While we have not discussed this is in the paper, our research does form part of a larger research initiative with Liverpool John Moores University, where such problems are being addressed [61]. Using the Mobile Agent Topology Test System (MATTS), compositions are analyzed based on their dependencies. While the technique can be applied to networked appliances, the engine is still being developed, and integrating this with the approach taken in this paper will be the focus of future work.

VIII. CONCLUSIONS

Recent years have seen an exponential growth in the use of home networks, from the Internet-enabled PC to network-enabled home appliances. Ordinary and everyday home appliances or devices are expected to become integral components of these networks. They will be required to join, leave, and self-configure themselves in accordance with their dynamic environment. While this brings considerable benefits, little progress has been made. Numerous systems have been presented in this paper that attempt to address the issue of interoperability. While each approach does provide benefits, they also suffer from drawbacks. By harnessing the positive aspects of these approaches, new research strands can be developed where devices can be seamlessly interconnected and operated. This is important because research conducted in isolation will not completely solve all interoperability concerns. Building on this requirement, we have described a possible scheme where a multidisciplinary approach has been adopted to incorporate the successful advances existing approaches have made. In doing this, we have been able to illustrate how new perspectives can be applied to device interoperability, function utilization, ontology usage, and device and service management.

The proposed framework was not designed to replace existing approaches but rather underpin them to reduce the amount of effort required by the user when defining tasks and configurations. Its real strength is the collaborative interactions between ad hoc devices and the functions they provide. This is a technique that has been successfully applied to other research domains such as peer-to-peer multimedia networks. By describing and processing semantic information in large distributed environments, devices can learn patterns to automatically form relationships with each other. By harnessing this interaction, emergent behavior can surface whereby high-level applications are created through low-level compositions.

Nonetheless, our approach is in the initial stages, and we acknowledge that more research is still required. While our working prototype demonstrates that it is possible to develop such a framework, its full effectiveness remains to be tested in a real-world situation. It does, however, suggest a line of enquiry that is likely to prove fruitful and shows that early experimentation has been positive. ■

Acknowledgment

The authors would like to thank the anonymous reviewers for providing highly constructive reviews, without which the paper would not have reached its current form. The authors would especially like to thank Dr. D. Llewellyn-Jones and Dr. J. Haggerty for reading the manuscript and providing constructive criticism.

REFERENCES

[1] S. Moyer, D. Marples, S. Tsang, and A. Ghosh, "Service portability of networked appliances," *IEEE Commun. Mag.*, vol. 40, no. 1, pp. 116–121, 2000.

[2] S. Kim and B. Zhang, "Genetic mining of HTML structures for effective web-document retrieval," *Appl. Intell.*, vol. 18, no. 3, pp. 243–256, 2003.

[3] G. Salton, J. Allan, and C. Buckley, "Automatic structuring and retrieval of large text files," *Commun. ACM*, vol. 37, no. 2, pp. 97–108, 1994.

[4] K. Fujii and T. Suda, "Dynamic service composition using semantic information," in *Proc. 2nd Int. Conf. Service Oriented Computing*, New York, 2004, pp. 39–48.

[5] Z. Song, Y. Labrou, and R. Masuoka, "Dynamic service discovery and management in task computing," in *Proc. IEEE 1st Int. Conf. Mobile Ubiquitous Syst.: Network. Services*, Boston, MA, 2004, pp. 310–318.

[6] R. Masuoka, Y. Labrou, B. Parsia, and E. Sirin, "The semantic web—Ontology-enabled pervasive computing applications," *IEEE Intell. Syst.*, vol. 18, no. 4, pp. 68–72, 2003.

[7] R. Masuoka, B. Parsia, and Y. Labrou, "Task computing—The semantic web meets pervasive computing," in *Proc. 2nd Int. Semantic Web Conf. (ISWC2003)*, Sanibel Island, FL, 2003, pp. 866–881.

[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic Web," *Sci. Amer.*, vol. 284, no. 5, pp. 34–43, 2001.

[9] R. A. Brooks, "The Intelligent Room Project," in *Proc. IEEE 2nd Int. Conf. Cogn. Technol. Human. Inf. Age*, Washington, DC, 1997, pp. 271–278.

[10] S. Peters and H. E. Shrobe, "Using semantic networks for knowledge representation in an intelligent environment," in *Proc. IEEE 1st Int. Conf. Pervasive Comput. Commun.*, Dallas-Fort Worth, TX, 2003, pp. 223–229.

[11] B. Rose, "Home networks: A standards perspective," *IEEE Commun. Mag.*, vol. 39, no. 12, pp. 78–85, 2001.

[12] O. Abuelma'atti, M. Merabti, and B. Askwith, "Internetworking the wireless domain," in *Proc. 3rd Int. Symp. in Commun. Syst., Networks Digital Signal Process. (CSNDSP)*, Staffordshire, U.K., 2002, pp. 344–348.

[13] T. Zahariadis and K. Pramataris, "Multimedia home networks: Standards and interfaces," *Comput. Standards Interfaces*, vol. 24, no. 5, pp. 425–435, 2002.

[14] S. Pattenden, P. Colebrooka, S. Ungar, F. Borghese, C. Francon, and R. Ambrosio, *Architecture for HomeGate, the residential gateway (AHRG)*, 2001.

[15] HAVI, "HAVI, the A/V digital network," San Mamon, CA, 2003.

[16] D. Evans, "In-home wireless networking: An entertainment perspective," *IEEE Electron. Commun. Eng.*, vol. 13, no. 5, pp. 213–219, 2001.

[17] G. Bhatti, Z. Sahinoglu, K. A. Peker, J. Guo, and F. Matsubara, "A TV-centric home network to provide a unified access to UPnP and PLC domains," in *IEEE 4th Int. Workshop Networked Appliances (IWNA)*, Gaithersburg, MD, 2002, pp. 234–242.

[18] P. Marshall, "Home networking: A TV perspective," *IEEE Electron. Commun. Eng.*, vol. 13, no. 5, pp. 209–212, 2001.

[19] C. Koumpis, L. Hanna, M. Anderson, and M. Johansson, "Wireless industrial control and monitoring beyond cable replacement," in *Proc. 2nd Profibus Int. Conf.*, Warwickshire, U.K., 2005.

[20] M. Minoh and T. Kamae, "Networked appliances and their peer-to-peer architecture AMIDEN," *IEEE Commun. Mag.*, vol. 39, no. 10, pp. 80–84, 2001.

[21] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proc. IEEE 1st Int. Conf. Peer-to-Peer Computing (P2P'01)*, Linkoping, Sweden, 2001, pp. 99–100, IEEE.

[22] K. Yang and J. Ho, "Proof: A DHT-based peer-to-peer search engine," in *Proc. IEEE Int. Conf. Web Intell.*, Hong Kong, China, 2006, pp. 702–708.

[23] A. Mingkhwan, P. Fergus, O. Abuelma'atti, M. Merabti, B. Askwith, and M. Hanneghan, "Dynamic service composition in home appliance networks," *Multimedia Tools and Applications (Special Issue on Advances in Consumer Communications and Networking)*, vol. 29, no. 3, pp. 257–284, 2006.

[24] P. Fergus, M. Merabti, M. B. Hanneghan, A. Taleb-Bendiab, and A. Mingkhwan, "A semantic framework for self-adaptive networked appliances," in *Proc. IEEE Consum. Commun. Network. Conf. (CCNC'05)*, Las Vegas, NV, 2005, pp. 229–234.

[25] L. Gong, "JXTA: A network programming environment," *IEEE Internet Comput.*, vol. 5, no. 3, pp. 88–95, 2001.

[26] B. Traversat, M. Abdelaziz, and E. Pouyoul. (2003). *Project JXTA: A Loosely-Consistent DHT Rendezvous Walker*, Sun Microsystems Inc. White Paper. [Online]. Available: http://research.sun.com/spotlight/misc/jxta-dht.pdf

[27] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–155, 1996.

[28] K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan, "Dynamic discovery and coordination of agent-based semantic web services," *IEEE Internet Comput.*, vol. 8, no. 3, pp. 66–73, 2004.

[29] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *Proc. 1st Int. Semantic Web Conf. (ISWC)*, Sardinia, Italy, 2002, pp. 333–347.

[30] M. Paolucci, K. Sycara, and T. Kawamura, "Delivering semantic web services," in *Proc. 12th Int. World Wide Web Conf.*, Budapest, Hungary, 2003, pp. 111–118.

[31] S. R. Ponnekanti and A. Fox, "SWORD: A developer toolkit for web service composition," in *Proc. 11th Int. World Wide Web Conf.*, Honolulu, HI, 2002, pp. 1–18.

[32] L. Chen, N. R. Schadbolt, C. Goble, F. Tao, S. J. Cox, C. Puleston, and P. Smart, "Towards a knowledge-based approach to semantic service composition," in *Proc. 2nd Int. Semantic Web Conf. (ISWC2003)*, Sanibel Island, FL, 2003, pp. 319–334.

[33] K. Fujii and T. Suda, "Component service model with semantics (cosmos): A new component model for dynamic service composition," in *Proc. 2004 Int. Symp. Applicat. Internet Workshops (SAINT 2004)*, 2004, pp. 321–327.

[34] J. Allen, *Natural Language Understanding*, 2nd ed. ser. Series Natural Language Understanding. Redwood City, CA: Benjamin/Cummings, 1994, p. 654.

[35] Z. Song, R. Masuoka, J. Agre, and Y. Labrou, "Task computing for ubiquitous multimedia services," in *Proc. ACM 3rd Int. Conf. Mobile Ubiquitous Multimedia*, 2004, pp. 257–262.

[36] Z. Song, Y. Labrou, and R. Masuoka, "Dynamic service discovery and management in task computing," in *Proc. IEEE 1st Int. Conf. Mobile Ubiquitous Syst.: Network. Services*, Boston, MA, 2004, pp. 310–318, IEEE Computer Society.

[37] E. Sirin, J. A. Hendler, and B. Parasia, "Semi-automatic composition of web services using semantic descriptions," in *Proc. 1st Workshop Web Services: Model., Architect. Infrastruct. (WSMAI'03)*, Angers, France, 2003, pp. 17–24.

[38] M. H. Butler, "Using capability profiles for appliance aggregation," in *Proc. 5th IEEE Int. Workshop Networked Applicat.*, Liverpool, U.K., 2002, pp. 12–16.

[39] D. Milojicic, P. Benadat, R. Corben, I. Greenberg, R. Kumar, A. Messer, D. Muntz, E. O. Strain, V. Poladian, and J. Rowson, "Appliance aggregation architecture," HP Laboratories, Bristol, U.K., Tech. Rep. HPL-2003-140, 2003.

[40] D. Chakraborty and A. Joshi, "Anamika: Distributed service composition architecture for pervasive environments," in *Proc. 9th Annu. Int. Conf. Mobile Comput. Network. (MobiCom)*, San Diego, CA, 2003, pp. 38–40.

[41] Y. Miwa. (2004). *Home appliances get connected*. [Online]. Available: http://www.ipv6style.jp/en/netapplnc/20041022/index.shtml

[42] I. E. T. Force, Ed., *Simple Service Discovery Protocol/1.0*. Reston, VA: Internet Engineering Task Force, 1999, pp. 1–18.

[43] P. Fergus, A. Mingkhwan, M. Merabti, and M. Hanneghan, "DiSUS: Mobile ad hoc network unstructured services," in *Proc. Personal Wireless Commun. (PWC'2003)*, Venice, Italy, 2003, pp. 484–491.

[44] Z. Yuejun and W. Mingguang, "Design of wireless remote module in X-10 intelligent home," in *Proc. IEEE Int. Conf. Ind. Technol.*, Hong Kong, China, 2005, pp. 1349–1353.

[45] OWL-S, 1.0. (2003). DAML. [Online]. Available: http://www.daml.org/services/owl-s/1.0/

[46] P. Fergus, A. Mingkhwan, M. Merabti, and M. Hanneghan, "Distributed emergent semantics in P2P networks," in *Proc. Inf. Knowl. Sharing (IKS'2003)*, Scottsdale, AZ, 2003, pp. 75–82.

[47] L. M. Stephens and M. N. Huhns, "Consensus ontologies—Reconciling the semantics of web pages and agents," *IEEE Internet Comput.*, vol. 5, no. 5, pp. 92–95, 2001.

[48] N. F. Noy and M. A. Musen, "PROMPT: Algorithm and tool for automated ontology merging and alignment," in *Proc. 17th Nat. Conf. Artif. Intell. (AAAI'00)*, Austin, TX, 2000, pp. 450–455.

[49] P. Mitra, G. Wiederhold, and M. L. Kersten, "A graph-oriented model for articulation of ontology interdependencies," in *Proc. 7th Int. Conf. Extending Database Technol.*, Konstanz, Germany, 2000, pp. 80–100.

[50] R. Kumar, V. Poladian, I. Geenberg, A. Messer, and D. Milojicic, "Selecting devices for aggregation," in *Proc. 5th IEEE Workshop Mobile Comput. Syst. Applicat. (WMCSA'03)*, Monterey, CA, 2003, pp. 150–159.

[51] J. Liu and V. Issarny, "QoS-aware service location in mobile ad-hoc networks," in *Proc. IEEE Int. Conf. Mobile Data Manage. (MDM'04)*, Berkeley, CA, 2004, pp. 224–235.

[52] P. Fergus, M. Merabti, M. B. Hanneghan, and A. Taleb-Bendiab, "Controlling networked devices in ubiquitous computing environments using biofeedback," in *Proc. 5th Annu. Postgraduate Symp. Converg. Telecommun., Network. Broadcast.*, Liverpool, U.K., 2004, pp. 91–96.

[53] M. Merabti, P. Fergus, D. Llewellyn-Jones, and P. Miseldine, "A seamless plug-and-play architecture for networked medical devices," in *Proc. Joint Workshop High Confidence Med. Devices, Software, Syst. (HCMDSS) Med. Device Plug-and-Play (MDPnP) Interop.*, Boston, MA, 2007, pp. 1–27.

[54] S. Castano, F. S. Montanelli, E. Panani, and G. P. Rossi, "Ontology-addressable contents in P2P networks," in *Proc. 1st Workshop Semantics Peer-to-Peer Grid Comput./12th Int. WWW Conf.*, Budapest, Hungary, 2003, pp. 55–68.

[55] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "EDUTELLA: A P2P networking infrastructure based on RDF," in *Proc. 11th Int. Conf. World Wide Web*, Honolulu, HI, 2002.

[56] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe, "A practical guide to building OWL ontologies using the protege-OWL plugin and CO-ODE tools edition 1.0," Univ. of Manchester/Univ. of Stanford, Manchester, U.K., 2004.

[57] V. Haarslev and R. Moller, "Description of the RACER system and its applications," in *Proc. Int. Workshop Description Logics*, Stanford, CA, 2001, pp. 131–141.

[58] B. McBride, "Jena: Implementing the RDF model and syntax specification," in *Proc. 2nd Int. Workshop Semantic Web (SemWeb'2001)*, Hong Kong, China, 2001, pp. 23–28.

[59] J. Haggerty, Q. Shi, P. Fergus, and M. Merabti, "Data authentication and trust within distributed intrusion detection system inter-component communications," in *Proc. 1st Eur. Conf. Comput. Network Defence (EC2ND'05)*, U.K., 2005, pp. 197–206, Univ. of Glamorgan.

[60] M. Kolberg, E. H. Magill, and M. Wilson, "Compatibility issues between services supporting networked appliances," *IEEE Commun. Mag.*, vol. 41, no. 11, pp. 136–147, 2003.

[61] D. Llewellyn-Jones, M. Merabti, Q. Shi, R. A. Askwith, and D. Reilly, "Improving interoperation security through instrumentation and analysis," in *Proc. 2nd Int. Workshop Interop. Solutions Trust, Security, Policies QoS Enhanced Enterprise Syst. (IS-TSPQ'06)*, Bordeaux, France, 2006, pp. 225–236.

ABOUT THE AUTHORS

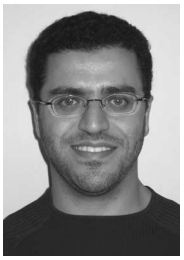
Madjid Merabti received the Ph.D. degree from Lancaster University, U.K.

He is Director and Head of Research, School of Computing and Mathematical Sciences, Liverpool John Moores University (JMU), U.K. He has more than 15 years' experience in conducting research and teaching in distributed multimedia systems (networks, operating systems, computer security). He has more than 90 publications in these areas. He leads the Distributed Multimedia Systems Group, which has a number of government- and industry-supported research projects in the areas of multimedia networking, differential services networking, mobile networks, networked appliances, sensor networks, intrusion detection, and network security architectures. He is collaborating with a number of international colleagues in the above areas. He is Co-Editor-in-Chief of *Pervasive Computing and Communications* and a member of the Editorial Board of *Computer Communications Journal*.

Prof. Merabti is Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA.

Paul Fergus received the B.Sc. degree (Hons.) in artificial intelligence from Middlesex University, U.K., the postgraduate diploma and M.Sc. degree in computing for commerce and industry from The Open University, Milton Keynes, U.K., and the Ph.D. degree in software engineering from Liverpool John Moores University (JMU), U.K.

He is a Research Fellow with JMU. He worked in industry for several years as a Software Engineer and has been directly involved in the development of several national and international projects. His research interests include networked appliances, home entertainment systems, wireless communications, peer-to-peer technologies, artificial intelligence, and the semantic Web.



Omar Abuelma'atti received the B.Sc. (Hons.) degree in electrical and electronics engineering from Ajman University, UAE, the M.Sc. degree in data telecommunications and networking from the University of Salford, U.K., and the Ph.D. degree in wireless networked appliances interoperability from Liverpool John Moores University (JMU), U.K.

He is a Lecturer at JMU, where he also is Network and System Administrator for the Networked Appliances Laboratory. His research interests include networked appliances, wireless and sensor networks, embedded Internet and electronic systems, and networked entertainment systems.

Heather Yu received the Ph.D. degree from the Electrical Engineering Department, Princeton University, Princeton, NJ, in 1998.

She was with Panasonic Princeton Laboratory from 1998 to 2007. In 2007, she joined the America Core Network R&D Department, Huawei Technologies, as Manager of Media Technologies. Her major research interests include multimedia communications, next-generation multimedia networks, peer-to-peer networking, and multimedia security. She has received 30 patents and has published one book, five book chapters, and more than 60 technical papers in related fields. Currently, she is Associate Editor-in-Chief of *Peer-to-Peer Networking and Applications Journal*.

Dr. Yu is Secretary of the IEEE P2P Communications and Networking Technical Subcommittee and a Voting Member of the IEEE Communications Society Strategic Planning Committee and Emerging Technologies Committee. She was Chair of the IEEE Multimedia Communications Technical Committee (2003–2006) and has been Technical Program Chair for many IEEE conferences.

Charlie Judice (Fellow, IEEE) received the Ph.D. degree in computational physics from the Stevens Institute of Technology, Hoboken, NJ.

He has joint appointments as an Eastman Fellow with Kodak, Rochester, NY; Visiting Research Professor at Rutgers—The State University, New Brunswick, NJ; and Adjunct Professor with Monmouth University, West Long Branch, NJ. He was Director of the Networked Imaging Technology Center, Kodak, where he lead an innovation team researching digital storytelling applications and technology before joining Kodak in 1996. He founded Verizon's Center for Networked Multimedia. From 1983 to 1996, he was Executive Director of Telcordia's Speech and Image Processing Research Department, Morristown, NJ. He led research on MPEG encoding, video on demand, ADSL, HDTV coding, image communications, speech processing, and many multimedia applications. He is currently Chairman of CAIP's Industrial Advisory Board and formerly a Member of Georgia Tech's College of Computing Advisory Board and Broadband Institute. He has received more than ten patients. He is author of more than 50 technical papers and several chapters in technical reference books. Currently, his research interests include digital storytelling, information fusion, sensor networks, and homeland security.

Dr. Judice is a member of Sigma Xi and Sigma Pi Sigma. He is an Editor of the IEEE MULTIMEDIA MAGAZINE and past Chairman of the IEEE Multimedia Committee, IEEE Communications Society.