XTalk: a Middleware for personalized service discovery in Future Internet

Preston Rodrigues, Soraya Ait Chellouche,Yérom-David Bromberg, Laurent Réveillère, and Daniel Négru
LaBRI – University of Bordeaux

***Abstract*—**
Future Internet is envisioned to cater to a wide range of multimedia services allowing their access through heterogeneous devices like laptops, TVs, PDAs and 3G mobile phones and interconnected via wired and wireless networking technologies. Such a heterogeneous environment (device and network) reinforces the need of a system that enable context-aware provisioning of multimedia resources to realize the some of the needs of FI like personalized media services and better end-user experience.

In this paper, we present XTalk : a context-aware multimedia service provisioning middleware. The XTalk middleware make use of a wide range of context information modeled using ontology and uses policies to automate the process of service discovery and service personalization.

I. INTRODUCTION

Over the last few years, with the improvements in communication technologies; anytime, anywhere *connectivity for anyone* has rapidly changed to *connectivity for anything*. This has given rise to a new envision of a global infrastructure connected by different devices, and commonly known as Internet of Things (IoT). In this era of IoT, devices travel along with their users and form new pervasive infrastructures consisting of both fixed and mobile heterogeneous devices. However, these devices have not been designed to interact seamlessly with each other. For instance, a user cannot make directly available the video captured by his camera to any member of his family independently of their devices' characteristics, their network locations and technologies.

The current trend of social networking and user generated multimedia content has been heavily responsible for a substantial increase in multimedia services available over the Internet. One way to access these services is to classify devices as either service providers or consumers. In either case, to provide seamless interaction, these devices need a mechanism to discover any available services in its local environments. Further, by utilizing the current context information, these devices can adapt to consumer situations and provide personalized services with better Quality of Experience (QoE) to the consumer. To this end, Service Discovery Protocols (SDPs) have been very popular and effective. However, they lack the means to utilize the available context information. Furthermore, SDPs such as SLP [1], WS-Discovery [2], and UPnP [3] do not allow service providers to be discovered and accessed anymore if they move from one network to another.

Towards this objective, we have identified three main issues that current SDPs should resolve to provide context-aware service discovery in the large: (i) service identification, (ii) service interoperability, and (iii) personalized service aggregation. Service identification becomes an issue since available services may appear back and forth across different networks. A service must be identified during its life cycle independently of the network it belongs to at a given time. However, information provided by current SDPs in request/response mechanism is not sufficient to uniquely identify a service. Furthermore, since Future Internet is seen as the aggregation of highly heterogeneous networks scattered around the globe, different SDPs may be used simultaneously. For instance, for each aggregated network, a different SDP may be used, making service interoperability an issue. Moreover, due to high consumer expectations for quality services, discovering remote multimedia services across many networks may produce responses, that may need to be adapted according to consumer and network preferences. Hence, different strategies should be explored and took in charge by the underlying infrastructure to provide consumers the most adequate and personalized services according to their preference and context information such as geographic locations, end-to-end delays, or available bandwidth. Therefore, personalized service aggregation becomes an issue.

In this paper, we make the following contribution

- We propose the XTalk middleware which enhance the core components proposed in ZigZag middleware architecture [4] to enable context-awareness.
- We introduce two new building blocks namely; the context manager and the policy framework. Their aim is to utilize the context information and operate filtering and aggregation strategies using policies to enables service consumers to find the most adequate remote services according to their search criteria and preferences.

The rest of this paper is organized as follows. Section II introduces our architecture. Section III highlights importance of context in Future Internet. Section IV presents the XTalk middleware. Section V discusses the related works. Finally, Section VI concludes and presents future work.

II. ARCHITECTURE

The continuous and rapidly growing popularity of multimedia services along with high consumer expectations for quality personalized services, places stringent requirement on the Internet. Currently, IP networks have to scale fully to handle ubiquitous high quality multimedia services. To ease this burden on the Internet, industry in collaboration with researchers has suggested several new architectures to redesign

978-1-4673-2781-7/12/$31.00 ©2012 IEEE

the Internet of the future [5] [6] [7]. One common design decision is to promote an architectural split into two main planes, decoupling services from transport infrastructure. More precisely, one plane is dedicated to upper layers to provide functions that control and manage service resources for service providers and consumers. The other plane is dedicated to lower layers to provide functions that control and manage transport resources to carry out data exchanges among service providers and consumers across heterogeneous networks. This architectural split enables functions dedicated to services and the ones dedicated to transport to evolve separately and independently. As a result, Future Internet offers users unrestricted access to quality personalized services outside their own network boundaries. However, this opportunity raises an issue related to personalized service discovery as envisioned in Future Internet project [8]. As service personalization and media adaption heavily relies on context information. To enable personalized service discovery in Future Internet we need to manage the heterogeneity among SDPs support by both service providers and service consumers.

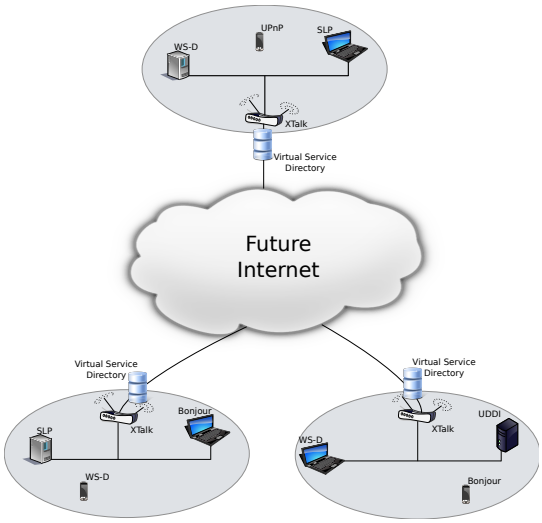

Fig. 1. XTalk Approach

To overcome the issues of personalized service discovery in Future Internet, we introduce the XTalk middleware. XTalk has been designed to enrich the upper layers proposed by Future Internet architectures while leveraging on the facilities available at the lower network layers. As illustrated in Figure 1, an instance of XTalk is deployed in each isolated local area network. Each network has one or more service providers and/or service consumers supporting different SDPs running simultaneously. The XTalk nodes monitor their own network and maintain a list of SDPs currently used by different service provides. This enables the XTalk nodes to uniquely identify service advertisements from consumer requests. Further, it uses the SDP list to detect and identify all the media services available within their network. Moreover, the XTalk nodes also keep track of the context information within their isolated local area network that enables them to personalize services based on the current context and service consumer search request. In order to promote the most popular services each XTalk node maintains a shared context-aware cache of most consumed services. This shared cache acts as a virtual service directory and is used by the XTalk nodes to search for all available services. All XTalk nodes communicate together thanks to the functions provided by the underlying Future Internet project [8].

When an XTalk node detects a request from a service consumer, it forwards the request to remote XTalk nodes. Based on the current user context and preferences, XTalk nodes then instantiate appropriate connectors to translate requests and responses from the protocol used by the service consumer to the one used by the service provider and vice versa. Once responses are received by XTalk node for a particular service request, they are personalized and aggregated prior to being returned to the consumer. For instance, if a remote XTalk node is not able to perform the needed translation, the translation request is forwarded to the most able XTalk node in the network. The functionalities performed by the XTalk nodes to provide personalized services are controlled and accomplished with the help of context management and policy framework.

The next section explains the importance of context information for personalization of services.

III. CONTEXT MODELING

With the emergence of mobile computing in the 90's, utilizing context information and designing context models have gained a lot of attention. The main purpose of context models are to analyze the current situation a user is in, and provide the user with personalized services that can adapt to his current context. Context can be users' personal characteristics (preferences, age etc.,), current location, time and device or network capabilities. A good context model helps to reduce the complexity of context-aware applications and improve their maintainability. However, defining and modeling context information is a complex and difficult task, since gathering of context information relies on combination of different sensory data and no formal way to describe relation among them. To this end, Ontologies provides the means to model a domain with the definition of objects and/or concepts, their properties and relations.

A. Ontology-based context model

Ontology provides a formal description and semantic for context information in term of objects, concepts, properties and relations. Hence, is a widely accepted tool for modeling context information in pervasive computing domain. The main reason for their acceptance is the popularity and maturity of Semantic Web Languages. For our Ontology based context model we choose Web Ontology Language (OWL) [9]. OWL has been successfully used by developer for applications needing a classification hierarchy, simple constrain feature and maximum expressiveness without losing any computational

completeness. In our model we identified 4 interrelated entities that are generic to any domain: namely;(i) the user, (ii) device, (iii) network, and (iv) service. These entities are modeled as `owl:Class` elements and the relations as `owl:ObjectProperty`. Each `owl:Class` is characterized by different `owl:DataProperty` that are considered relevant to the domain.

The `User` entity is described in different profiles:

- The `GeneralProfile` contains general information about the user such as name, age, etc..
- The `SubscriptionProfile` contains information on the different services for which the user have subscribed and the services that he may access.
- The `ContactProfile` contains the contact information of the user such as his address, phone number, SIP URI, etc.
- The `Affiliationprofile` contains information about the different organization to which the user is affiliated.
- The `AuthenticationProfile` contains information that allows the user to be authenticated.
- The `PreferenceProfile` contains the user-defined preferences or the deduced preferences from usage. The user preferences could be generic and applied to any service or situation or they could target a specific service or context entity and thus be applied only when the latter is involved.

The `Device` entity is described as two sub-entities namely: (i) `HardwarePlatform` and (ii) `Softwareplatform`. The `HardwarePlatform` entity is modeled in a hierarchical way since the components can be either atomic or composite. On the other hand the `Softwareplatform` entity represents User and System software's that runs on the device. For example, audio/video codec or players.

The description of the `Network` entity comprises of information such as the name of the network and the theoretical parameters that characterize it. For instance, user location or a loss or error rate experienced in a multimedia session within a related network. These parameters are either reported by monitoring modules or evaluated using subjective techniques.

The `Service` entity represents the different services that the user can access. Any entity that provides a service is represented within this entity. The service entity is modeled using OWL-S [10] Service Profile that relies on Inputs, Outputs, Preconditions and Effects (IOPEs) as modeling parameters.

B. context-aware service personalization and adaptation

The main advantage of service personalization and adaptation is to provide to the user the optimal variant of content based on the current context information in its local environment. To this end, we have developed XTalk: a context-aware policy based middleware. In the example below we show how policy is enforced to accomplish a particular task.

The following policy selects the category and video resolution according to the user preferences, location and device. The policy states that when user John is at his office premises, he prefers to get the match highlights of recently played football matches when he requests for a VoD service. As John's office has a very good network bandwidth the policy selects the High Definition (HD) resolution video for John. The simple policy that is generated using John's current context information for HD content streaming is shown below.

```
User(John) ^ Service(VoD) ^ location
(fixed, Office) ^ Device(Laptop)
^ PrefProfile(John, Office)^
VideoStart(00:00) ⇒ allow(Football
highlights,HD)
```

However, user John now wants to continue viewing the same VoD service from his mobile device on this way home. He now logs in from his mobile phone and accesses the same VOD service. At this point the network entity detects the John's bandwidth is not enough to consume HD content and hence updates the policy to stream SD resolution optimized from mobile viewing. Furthermore, the video starts to stream from the same position that John paused on the office computer. The updated policy is shown below.

```
User(John) ^ Service(VoD) ^ location
(mobile, travel) ^ Device(Phone)
^ PrefProfile(John, travel) ^
VideoStart(3:12) ⇒ allow(Football
highlights,SD_M)
```

The next section explains in details our context-aware policy based XTalk middleware.

IV. The XTalk middleware

We now present an overview of ZigZag middleware highlighting its functional and architectural advantage in enabling service discovery in the large.

A. ZigZag middleware Overview

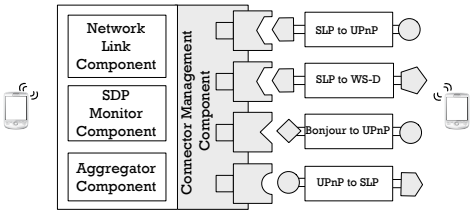

Fig. 2. ZigZag Middleware

The ZigZag middleware [4] enables SDPs initially designed for local area networks to work across highly heterogeneous networks targeting the needs of Future Internet. ZigZag approach is based on protocol translation to enable service discovery irrespectively of their underlying SDP. As illustrated in Figure 2, the ZigZag middleware is architectured around 4 core components, namely: (i) a *SDP Monitor Component* to detect the current SDPs being used, (ii) a *Connectors Management*

Component to instantiate the adequate SDP translator, (iii) a *Network Link Component* to maintain connections among ZigZag nodes, and (iv) an *Aggregator Component* to apply aggregation strategies. The core functionalities of each component are briefly explained below.

SDP Monitor Component: The *SDP monitor* checks the availability of different SDPs in once local environment. The SDP monitor is designed to identify all SDPs currently used in the network. Furthermore, it can uniquely identify service advertisements from consumer requests, this enables the SDP monitor to locally cache service advertisements and map them to a Universally Unique IDentifier (UUID) so that services can be identified uniquely across different ZigZag nodes.

Connectors Management Component: A *Connector* translates one SDP to another SDP. It is specific to a pair of SDPs. Thus, there exists as many connectors as there exists different pair of SDPs between which interoperability is required. Currently, the *Connectors Management Component* supports on the fly instantiation of one or more z2z gateways [11] that act as connectors. We particularly choose z2z as provides an optimized runtime system and facilities for describing network protocol behaviors, message structures, and translation logic. Additionally, the Connectors Management Component collects statistics about SDPs being used to take in charge a fine grained life cycle of instantiated connectors. It can start, stop, pause or resume connectors according to the most often detected SDPs.

Network Link Component: ZigZag nodes are directly connected to each other irrespectively of the underlying network infrastructure supported by the Future Internet. *Network Link Component* implements a simple protocol for building a data distribution tree among ZigZag nodes enabling them to exchange multicast messages about discovered SDPs, and services across each isolated local area network.

Aggregator Component: The *Aggregator Component* collects a bunch of messages coming back and forth from several connectors instantiated by the Connectors Management Component. More specifically, the Aggregator Component accumulates all SDPs responses coming from different remote ZigZag nodes, and selects the one that matches best the criteria of the associated request, and then forwards it to the service requester.

The XTalk middleware leverages on the components of ZigZag and enhance them to provide context-awareness with the help of policies. The XTalk middleware architecture is explained in details in the following subsection.

B. XTalk Architecture

The context-adaptive XTalk middleware takes advantage of the modular component designed proposed in ZigZag middleware [4] and enhance them to utilize the state of the art context-awareness with the help of policies. The policies enable XTalk middleware to provide necessary media service adaptation for personalized services. The XTalk middleware architecture consists of 3 main components, namely: (i) the *context manager* (ii) the *policy framework* and (iii) the *core components*. As depicted in Figure 3, these component are plugged together to provide media adaptation by utilizing the available context information. Furthermore, these components work in tandem to provide the needed interoperability by utilizing the state of the art translation process so that users can consume highly personalized multimedia services. The core functionalities of each component and their sub-components are deeply explained below:

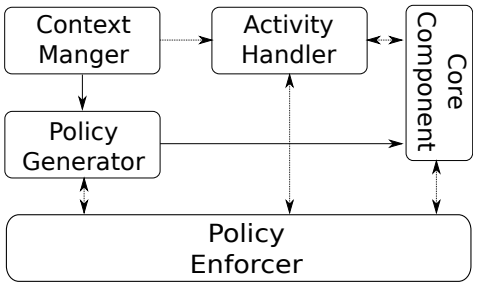

Fig. 3. XTalk Architecture

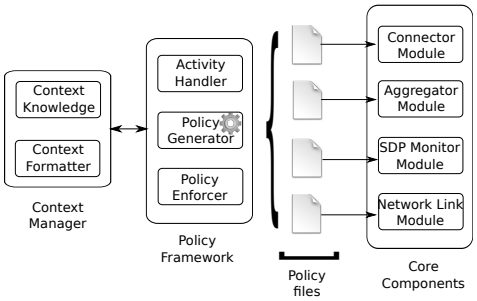

Fig. 4. XTalk Modules

1) Policy Framework: As depicted in Figure 4, the Policy Framework consists of 3 modules, namely: (i) the *activity handler* (ii) the *policy generator* and (iii) the *policy enforcer*. The functionalities of these modules are explained below.

Activity handler: The Activity handler is an event manager that monitors various events that can occur in the system at runtime. Furthermore, it is also used to define new events as required by the core component modules to enable new functionality. Further, it ensures that events are assigned proper priority during system runtime. For instance, if a request by a consumer has more then one response messages then the `aggregate` event is triggered. However, if this XTalk node is not able to translate the messages in order to aggregate them, it triggers a `system specific` event. In such a case the activity handler assigns a higher priority to this event (`system specific`). This helps the XTalk node to take advantage of the distributed translation mechanism supported by the middleware.

Policy generator: Is a dedicated compiler which can translate context information modeled using ontologies into the representations required to implement policies. The Policy generator, generates separate polices as required by different core component modules. These files are then dynamically

loaded by the concerned modules. Furthermore, it is also responsible for informing the policy enforcer about the any new generation of policy files so that the policy enforcer can update its process.

Policy enforcer: The policy enforcer has three main functionality. Firstly, it is used to describe the process by which a new dynamic service flow is created in the system upon a resource request. Secondly, it is used to validate the policies depending on the resources present in the system and finally it authorizes the process to enable a smooth system flow. For instance, if a consumer explicitly wants to view a HD resolution video content in SD resolution even though he has all the resources to view it in HD, the policy enforcer adheres to the consumer's preference and authorize the process.

2) Context Manager: The context manager consists of a *context formatter* and *context knowledge*. The *context knowledge* consists of generic ontology layer specific to a domain, in our case it supports multimedia services. On the other hand, the *context formatter* is in charge of formatting the raw context data acquired from the different context providers. The context providers supply context information as a simple XML file. The *context formatter* uses this XML file to generate a context model with the help of XML Stylesheet Language Transformation (XSLT) tools. This enables the *context formatter* to abstract the heterogeneity from the acquired context data. XSLT uses an XSLT transformation template to transform the XML file into a context model so that it can be utilized by other components of the middleware. To support other context description standards, a different XSLT transformation template can be loaded into the context formatter. Furthermore, the context manager is in-charge of the notification process that supplies the Context Consumers with needed context information. It is also responsible for maintaining the set of context providers that supply the context knowledge through sensing information.

V. RELATED WORK

The proliferation of SDPs to discover various services across different networks is the source of a major heterogeneity issue. Service consumers must be able to discover anytime anywhere remote services irrespectively of their underlying SDPs. Over the past decade, many solutions have emerged to provide interoperable service discovery such as ReMMoc [12], INDISS [13], z2z [11] and Starlink [14]. ReMMoc is a reflective middleware that provides a specific API to hide to applications the underlying SDPs currently used in the local network environment. However, ReMMoC require developers to redesign all existing applications to make them compliant with the ReMMoC API, which is quite a daunting task. This particular constraint is overcome with INDISS that is a transparent middleware that provides interoperability to existing applications without altering them. Although INDISS and ReMMoC could be considered as one step forward in the challenge of interoperable service discovery, these last years have seen two other approaches, z2z and Starlink, that have brought many facilities to enable transparent translation of one protocol to another. More precisely, z2z and Starlink provide an optimized runtime system and facilities for describing network protocol behaviors, message structures, and translation logics. Such facilities come from the fact that they rely on a high-level definition language that hides low level network details and highlights only key properties of protocols to translate.

With the maturity of Semantic Web Languages several ontology-based context models have been proposed. A Context Ontology Language CoOL [15] describes contextual facts and contextual interrelationships by projecting the conceptual based model Aspect-Scale-Context information (ASC) [16] to language elements. CONtext ONtology CONON [17] captures the general features of basic contextual entities like (location, user, activity and computational entity) and uses the domain specific ontology to describes the concepts related to a specific domain. While COBRA−ONT [18] is a collection of ontologies expressed in OWL for context-aware systems. A COBRA−ONT defines concepts associated with four distinctive but related themes: places, agents, agents' location and agents' activity. SOUPA [19], developed by the same authors as COBRA−ONT, deals with pervasive computing. SOUPA is composed on two parts: (i) the SOUPA core that holds ontologies which provide a common vocabulary for different pervasive computing environments and (ii) the SOUPA extension which contains ontologies for different domain specific vocabularies. To overcome the interoperability issue, SOUPA maps its concepts to the concepts of several common ontologies such as FOAF [20], DAMLTime [21], etc. The reported models have shown that ontologies, by enabling semantic interoperability and logic inference, offer the necessary means to build efficient context models.

Policies are goals that allow a system to complete a particular task. They have the tendency to influence the runtime behavior of system elements to change dynamically; without changing or re-starting the system. Therefore, policies have become a popular approach to cope with the increasing complexity of distributed system. Some examples of policy languages include ASL (Authorization Specification Language) [22] PDL (Policy Description Language) [23]. ASL is a formal logic language for specifying access control policies. It uses meta-policies called integrity rules to specify application dependent rules that enable it to limit the range of access control policies. Although it provides support for role-based access control, it does not scale well to large systems because there is no way of grouping rules into structures for reusability. This is overcome by PDL. PDL is an event-based language that uses the event-condition-action rule paradigm and consult active databases to define a policy as a function. These functions are then mapped to a series of events into a set of actions. The language has a clearly defined architecture and semantics. However, PDL does not support access control policies, nor does it support the composition of policy rules into roles.

VI. CONCLUSION

In this paper, we have identified three challenges that current SDPs need to resolve in order to provide personalized service discovery in highly heterogeneous networks like Future Internet. For this purpose, we have presented the XTalk middleware. The XTalk middleware uses context modeling approach and policies to describe the system flow. The context model uses ontology representation based on the basic context descriptors. These descriptors are considered as base classes in the ontology hierarchy to represent domain independent concepts. Further, the policies are used to dynamically change the behavior of modules at runtime without restarting the system.

While XTalk has provided a step forward, there are limitations to the approach that we will investigate further. As ongoing future work, we are currently investigating privacy, a key concern in personalized service discovery. Furthermore, we plan to integrate and configure privacy-specific behavior using policies within the XTalk middleware architecture.

Although validating the correctness of policies has not been proven. We have performed custom tests in-house to verify the outcome with the expected results. Additionally, we would like to validate the correctness of polices in a large scale deployment. Currently, we are in the process of deploying the XTalk middleware in a large scale media provisioning architecture ALICANTE. This large scale deployment will enable us to fully evaluate our XTalk middleware by providing valuable feedback to fine grain the policies. Furthermore, it will also help us to validate the correctness and performance of policies in a large scale network.

Acknowledgements

This work is supported by the European research project ALICANTE within the framework of the EU FP7 in ICT, under grant agreement No. 248652/ /ICT–ALICANTE/ http://www.ict-alicante.eu

REFERENCES

[1] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol (SLP)," *http://www.ietf.org/rfc/rfc2608.txt*. [Online]. Available: http://www.ietf.org/rfc/rfc2608.txt

[2] J. Beatty, G. Kakivaya, D. Kemp, and T. Kuehnel, "Web Services Dynamic Discovery (WS-Discovery)," *http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html*. [Online]. Available: http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01

[3] A. Presser, L. Farrell, D. Kemp, and W. Lupton, "Upnp device architecture 1.1," *www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf*.

[4] P. Rodrigues, Y. Bromberg, L. Réveillère, and D. Négru, "Zigzag: a middleware for service discovery in future internet," in *12th IFIP International Conference on Distributed Applications and Interoperable Systems*, ser. DisCoTec'12, 2012, pp. 208–221.

[5] "AKARI Architecture Design Project," *http://akari-project.nict.go.jp/eng/index2.htm*.

[6] "Future Networks Projects," *http://cordis.europa.eu/fp7/ict/future-networks/*.

[7] "NSF Future internet architecture project," *http://www.nets-fia.net/*.

[8] "ALICANTE: mediA ecosystem depLoyment through ubIquitous Content-Aware NeTwork Environments," *http://www.ict-alicante.eu/*.

[9] D. McGuinness, F. Van Harmelen *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, pp. 2004–03, 2004.

[10] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne *et al.*, "Owl-s: Semantic markup for web services," *W3C Member submission*, vol. 22, pp. 2007–04, 2004.

[11] Y. Bromberg, L. Réveillère, J. Lawall, and G. Muller, "Automatic generation of network protocol gateways," *Middleware 2009*, pp. 21–41, 2009.

[12] P. Grace, G. Blair, and S. Samuel, "ReMMoC: A reflective middleware to support mobile client interoperability," *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pp. 1170–1187, 2003.

[13] Y.-D. Bromberg and V. Issarny, "Indiss: Interoperable discovery system for networked services," in *IFIP/ACM/Usenix International Middleware Conference*, 2005, pp. 164–183.

[14] Y.-D. Bromberg, P. Grace, and L. Réveillère, "Starlink: Runtime interoperability between heterogeneous middleware protocols," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ser. ICDCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 446–455. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2011.65

[15] T. Strang, C. Linnhoff-Popien, and K. Frank, "Cool: A context ontology language to enable contextual interoperability," pp. 236–247, 2003.

[16] T. Strang, "Service interoperability in ubiquitous computing environments," Ph.D. dissertation, Ludwig-Maximilians-University Munich, Oct. 2003.

[17] X. Wang, D. Zhang, T. Gu, and H. Pung, "Ontology based context modeling and reasoning using owl," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. Ieee, 2004, pp. 18–22.

[18] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The Knowledge Engineering Review*, vol. 18, no. 03, pp. 197–207, 2003.

[19] H. Chen, F. Perich, T. Finin, and A. Joshi, "Soupa: Standard ontology for ubiquitous and pervasive applications," pp. 258–267, 2004.

[20] D. Brickley and L. Miller, "Foaf vocabulary specification 0.91," Tech. rep. ILRT Bristol, Nov. 2007. ur l: http://xmlns. com/foaf/spec/20071002. html, Tech. Rep., 2000.

[21] F. Pan and J. Hobbs, "Time in owl-s," in *Proceedings of AAAI-04 Spring Symposium on Semantic Web Services*, 2004.

[22] S. Jajodia, P. Samarati, and V. Subrahmanian, "A logical language for expressing authorizations," in *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*. IEEE, 1997, pp. 31–42.

[23] J. Lobo, "Apolicy description language," 1999.