Scalable And Interoperable Service Discovery For Future Internet

Preston Rodrigues, Laurent Réveillère, Yérom-David Bromberg, Daniel Négru

Scalable And Interoperable Service Discovery For Future Internet

Preston Rodrigues
CNRS / LaBRI
University of Bordeaux
Talence, France
preston.rodrigues@labri.fr

Laurent Réveillère
CNRS / LaBRI
University of Bordeaux
Talence, France
laurent.reveillere@labri.fr

Yérom-David Bromberg
CNRS / LaBRI
University of Bordeaux
Talence, France
david.bromberg@labri.fr

Daniel Négru
CNRS / LaBRI
University of Bordeaux
Talence, France
daniel.negru@labri.fr

ABSTRACT

We live in a highly networked world where users and their devices interact with other devices with the help of services. With the current trend of social networking and the engendering of user generated multimedia content through personal mobile devices, services are not just confined to enterprise servers. In fact, services travel along with users and/or devices that may act as both mobile service providers as well as service consumers. This trend paves the way for the next generation of Internet (Future Internet) where a very large number of such mobile devices will provide and consume ubiquitous services. The Future Internet (FI) is an opportunity to address service discovery combined with mobility enabling: (i) service providers to join networks and announce their services and (ii) service consumers to discover and invoke services irrespectively of their location. This paper focuses on service discovery for large scale network in the context of Future Internet, taking into account protocol interoperability and scalability.

Categories and Subject Descriptors

C.2.4 [**Distributed systems**]: [Distributed applications]; H.3.3 [**Information Search and Retrieval**]: [Search process, Selection process]

Keywords

Future Internet, Scalable Service Discovery,Interoperability

1. INTRODUCTION

M-MPAC'2011, December 12th, 2011, Lisbon, Portugal.

Since the design of the current Internet over 40 years ago, it has been under a constant strain from diverse and ever increasing services with stringent requirements. However, the current trend of social networking and personalized services [16] has lead researchers and industry alike to look for more favorable solutions to ease the burden. Moreover, user generated multimedia content and services using personal mobile devices [8] are gaining a lot of momentum, indicating an increase in production and consumption of personal multimedia content. As a result, researchers have suggested several new design paradigms to build the Internet of the future. Currently, many projects are working towards this objective, leading to a lot of ongoing research to redesign the current Internet architecture [1, 2, 3, 4].

Future Internet is presently seen as an evolution of current Internet focusing on content, services, users, mobility and their management instead of machines. However, there is no agreement on what the technology of the future Internet will manifest to. We assume that, in future Internet, service consumers will be able to discover anytime, anywhere, remote services irrespectively of their underlying Service Discovery Protocols (SDPs). For example, a user may wish to make the video captured by his camera directly available to any member of his family, whether they are within the same network or not. Therefore, any user can act as a service provider, making its own content available to others. Moreover, current network architectures are unsuitable for enabling such services to be discovered and accessed when the location of service provider and consumer and the network characteristics may change at any time.

Existing network architecture mainly supports local and Internet level service discovery, with no protocol supporting both. Hence, if a networked device acting as a service provider moves from one network area to another, it can not be discovered anymore. For instance, Service Discovery Protocols, such as SLP [15], WS-Discovery [7], and UPnP [21], assume that service providers never move and are thus not adequate for Future Internet. Some Internet level SDPs rely on an overlay network architecture built through the use of *trackers* and/or *Distributed Hash Tables (DHT)* to support

service discovery across different network areas. However, such SDPs must deal with high churn rates due to the fact that networked devices can join and leave the system at any time [24]. This issue remains a challenging problem in DHTs despite the considerable research efforts that have been made in the domain.

In Future Internet, devices will be able to provide services in any networked environment supporting different SDPs. Interoperability among these SDPs is a key issue as it must be handled across various network paths from service consumers to service providers and vice versa. To address this issue, we utilize the existing components from the Future Internet architecture proposed by an European Research project, ALICANTE [1] [2] and enrich them with a new service discovery mechanism. Our approach relies on network reflection to detect protocols currently used by devices, and dynamic instantiation of gateways according to the network context to provide the adequate interoperability among SDPs currently used. In this paper, we make the following contributions:

- We enhance an existing Future Internet architecture proposed in ALICANTE [2], to enable large scale service discovery. Our proposed architecture enables SDPs, initially designed for local area networks, to work across any network area.
- We provide a new interoperability system that seamlessly integrates with the architecture, to translate on the fly one SDP to another.
- We show the applicability of our approach by assessing a preliminary prototype of our interoperability system.

The rest of this paper is organized as follows. Section 2 discusses related work. In Section 3, we describe a service discovery mechanism for large scale networks as required by Future Internet. Section 4 describes the components of our interoperability system . Section 5 presents our prototype. Finally, Section 6 concludes the paper.

2. RELATED WORK

Service discovery is a mechanism that enables service consumers to find remote services without having any previous knowledge of neither their locations nor their characteristics. There are many Service Discovery Protocols (SDPs) available each having a specific discovery technique to discover services in the network. SDPs can be dedicated for local area networks or for large scale networks and can rely on either centralized, decentralized or hybrid service directories to store their service information. An extensive literature survey [6, 20] gives an in depth analysis of different service discovery protocols.

SDPs for local area networks. In a local area network, SDPs that do not rely on any central service directory, such as SLP [2] [15], WS-Discovery [3] [7] and UPnP [21], can be discovered using two different modes: active or passive. In an active mode, a service consumer that wants to interact with a specific service provider, needs to broadcast/multicast a search query on the network. Any service providers capable of providing the requested service, respond to this query with its location information. The service consumer then makes a unicast request to the service provider to retrieve the information necessary to consume the service. In a passive mode, a service provider broadcasts/multicasts service announcements into the network. A service consumer interested in a particular service: (i) first listens for its announcements, and then (ii), once it gets a corresponding announcement, makes a unicast request to the service provider to retrieve the information necessary to consume the service. Alternatively, the service consumer may also cache the service announcements for future use. Concerning SDPs that rely on a central directory to stores service information, such as Jini [17], Salutation [5] and Bonjour [11, 12], both service consumers and providers need to multicast requests to find an available service directory. Once it is done, on the one hand, service providers are able to register their services using a unicast request, and, on the other hand, service consumers can make a unicast request to finally search available services.

SDPs for large scale networks. Although the aforementioned SDPs have been very successful in home and enterprise networks, they do not scale well outside local area networks, and in particular in large scale networks. In fact, roughly, service consumers use multicasting to discover services which leads to network flooding. Hence the uses of such protocols are not recommended for large scale networks like, for instance, Internet. Over the past decade, trackers and Distributed Hash Tables (DHTs) (e.g., Chord [27], CAN [22], Pastry [25], Tapestry [25], Kademlia [19]) have been getting a lot of interest as a substrate to build overlay networks. Such networks provide particular protocols to store, search and distribute service information across the network overlays by either using overlay multicast range queries or use flooding and random walks to enable service discovery. These protocols have been very successful in large scale networks due to their self-organizing, fault tolerance and massive scalability properties. However, they introduce additional processing overhead: they use extra packet headers needed to manage the overlay. This creates a huge bottleneck for resource-constrained devices as they embed little memory and processing power. Furthermore, due to their unique way of storing and distributing service information, these protocols do not interoperate well with each other as well as with other existing service discovery protocols. Although UDDI [13] is designed for Internet level service discovery it assumes that the service provider and consumer already knows its location.

SDPs heterogeneity. The proliferation of SDPs to discover diverse 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. Further, service providers must be discoverable not only in their vicinity, but also outside their local area network, such as a large scale network as required by Future Internet. Interoperability among SDPs is not new in the context of local area network [10]. However, existing solutions do not address SDPs interoperabil-

[1] ALICANTE: Media Ecosystem Deployment Through Ubiquitous Content-Aware Network Environments

[2] SLP can also be configured to rely on a central service directory (Directory Agent).

[3] WS-D can dynamically configures itself to rely on a central service directory (Discovery proxy).

ity crossing local network boundaries to large scale networks and vise versa [10, 14]. Concerning SDPs based on overlay networks, interoperability has never been considered due to the very different nature of the DHTs used. To address the issue of heterogeneity, we propose an approach that enables local area SDP messages to be translated and forwarded in a progressive manner across large scale network paths to discover remote services whatever their physical location.

3. FUTURE INTERNET SERVICE DISCOVERY ARCHITECTURE

In order to overcome the wide variety and the ever growing number of heterogeneous protocols in the context of Future Internet (FI), we utilize an existing Future Internet architecture proposed by ALICANTE [2] based on a backbone that aims to federate multiple Home Network Environments (HNEs). This backbone inherits key principles from Service Oriented Architectures (SOA) and acts as a Service Network Environment (SNE). A SNE defines a set of loosely coupled self contained services by a service provider that use service directories to publish the available content and services (e.g. multimedia, QoS) to be searched and consumed by end user devices. As illustrated in Figure 1 ③ on one hand, HNEs are densely populated by different kind of devices such as smart phones, laptops, IPTV and cameras with either wired or wireless interfaces, whereas, on the other hand, SNEs are densely populated by services providers and directories. Service directories, inside a SNE, are organized as a federation of directories irrespective of their underlying service discovery protocols. Thus, directories enable service providers to publish seamlessly their services over various SNEs. Hence, services from SNE interact seamlessly with devices from HNEs and *vice versa* through the use of gateways that acts as an enabler. Such gateways, also called residential gateway (usually delivered by Internet Service Provider), connect HNE to SNE. In other terms, gateways help devices in one HNE to discover services offered by other devices that belong to another HNE.

We presume that, in Future Internet architectures interoperability will have to be accosted due to the constant interaction among diverse elements (devices, services, providers and directories) in different environments. In order to addresses this issue and enable seamless interaction, we designed an interoperability system that can be deployed on devices in HNEs, SNE, and/or gateways. Figure 1 highlights the different service discovery scenarios that can be encountered. In the first scenario (Figure 1 ①), devices, which have enough resources in terms of CPU and memory, can embed interoperability system to interact seamlessly with their immediate vicinity. Thus, devices can discover local services available in their related HNE in a quick manner, as no interactions with a SNE are required. On the contrary, in Figure 1 ②, devices, which have limited resources, may invoke interoperability system deployed on the gateway available in their HNE to discover available services (i.e. either local or remote) that rely on heterogeneous SDPs. Finally, in Figure 1 ③, interoperability system is required in a SNE to support seamless exchanges of service information within the federation of service directories whatever the underlying service discovery protocol being used.

To summarize, this architectural usage serves two main purposes:

1. End user devices in a HNE can discover services without needing to support all SDPs that are supported by different service providers in the SNE or in other HNEs.

2. Service directories joining the federation do not have to worry about how services will be discovered by service consumers, i.e. end user devices.

Joining or leaving the network anytime, anywhere, whatever the underlying protocols currently used is made possible through interoperability system. This interoperability system instantiate dynamically specific components that translate on-the-fly one protocol to another. The detailed architecture of the interoperability system is explained in the next section.

4. INTEROPERABILITY SYSTEM

In this section, we present our Interoperability system architecture that enables service discovery in a heterogeneous environment. Furthermore, we also present our response message aggregation strategy which helps to aggregated service information from different SDPs in the connected networks.

4.1 Architecture

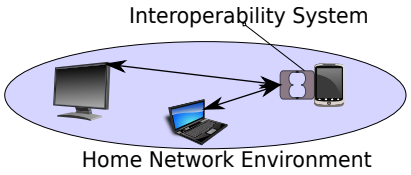
We designed an Interoperability System architectured around 5 core components, namely: *transport handler*, *SDP monitor*, *protocol selector*, *collaborator* and *connectors*. These components are plugged together to form a five steps translation process (See Figure 2). Each of the component and their functionalities are explained below:

Transport Handler. The *transport handler* is responsible for identifying uniquely different service consumers. It receives incoming SDP requests and sends related responses to the adequate service consumer.

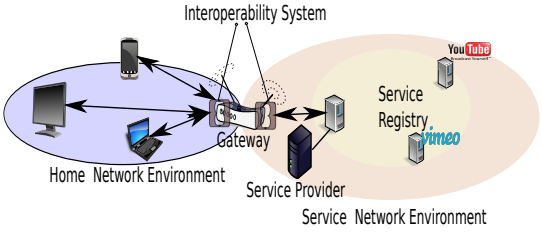
SDP Monitor. The *SDP monitor* is a network monitor responsible for checking the availability of different service discovery protocols in its environment. The SDP monitor is designed to keep track of SDPs currently used. It periodically checks the networked environment for the availability of new SDPs and maintains their response timeouts.

Connectors. 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. A connector may be a third party component. Currently, we leverage on the z2z interoperable system [9] to act as a connector. However, in our architecture, an interoperable service is not tightly bound to z2z, and may rely on any other translator.

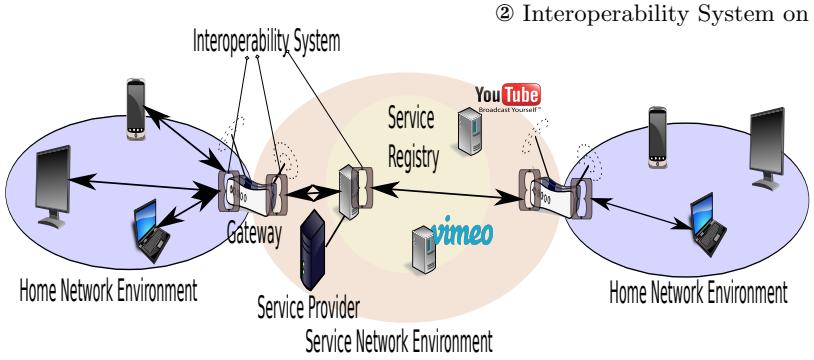
Protocol Selector. The *protocol selector* compares SDPs used by service consumers with SDPs used by service providers and take a decision about their compatibility. The compatibility criteria is defined in the policy file and takes into account message type,transport used (UDP,TCP) and message timeouts. Based on its decision, the adequate *connector* is instantiated *via* the *collaborator*.



① Interoperability System on Consumer device



② Interoperability System on Gateway Element



③ Interoperability System on Gateway Element and Service Registry

Figure 1: Future Internet: Service Discovery Scenarios with Interoperability System

Collaborator. Based on the decision made by the *protocol selector*, the *collaborator* instantiates and/or releases dynamically one or more connectors to translate one SDP to another for one or more service consumers.

Figure 2 depicts the message flow among the components during the service discovery process. A search request by a service consumer is span into as many search requests as there exists compatible SDPs in the networked environment. Correspond replies are then aggregated into a single response and sent back to the service consumer. For instance, in Figure 2, a service consumer that supports only SLP makes a SLP request to discover services in the network. The *transport handler* receives the initial incoming SDP search request. This request is then forwarded to the *protocol selector* that identifies the SDP used by this particular client (i.e. SLP). For each such request to the *protocol selector* an event is triggered to the *SDP monitor* to check for a list of available protocols in the network. Once this list is known by the *protocol selector*, it makes a decision as to what would be the most compatible protocols for the incoming request. This decision is then forwarded to the *collaborator*, which is responsible for instantiating the corresponding required *connectors*. In this example, the *protocol selector* decides that SLP is compatible with UDDI, UPnP and WS-Discovery SDPs, and therefore instantiates in parallel SLP-to-UDDI, SLP-to-UPnP, and SLP-to-WSDiscovery *connectors*. Once the collaborator receives replies from the instantiated *connectors*, it aggregates them into one single response. This helps make maximum use of data utilization in a message by increasing the amount of data concentrated in a single response message, thereby reducing message cost incurred during service discovery. The aggregated message is then sent to the *transport handler*, which in turn forwards the response to the relevant requesting service consumer.

4.2 Message Aggregation

In a request/response messaging paradigm the requesting entity has to wait for a fix amount of time before receiving a response for its request. If the response does not arrive before this fix time slot the requesting entity signals a timeout and resends the request. However, if the same request is sent to multiple nodes in parallel, response aggregation is considered as the preferred solution. Response message aggregation ensures that, several parallel responses of a particular request are combined into a single response message, thereby reducing the number of messages on the wire and preserving scarce bandwidth. Several papers [18, 23, 26] have shown the importance of messages aggregation in different network environments, but they rely on a single protocol with constant timeout. However, in the message flow example depicted in Figure 2, we employ an aggregation strategy for multiple service discovery protocols to aggregate response messages having different timeout.

Aggregation with timeout. For a response message to be valid for aggregation it must arrive before the source SDP signals a timeout. However, in a multi protocol service discovery scenario (example Figure 2) response messages may arrive after the source SDP timeout as different SDP have different timeouts. In order to make the responses that arrive after the timeout valid for aggregation we designed a message aggregation strategy. Our aggregation algorithm is explained below.

Message Aggregation Algorithm. Our Message aggregation algorithm 1 in the *collaborator* relies on the *Protocol Selector* to retrieve a list of compatible service providers SDP in the network and their timeout. The *Protocol Selector* makes sure that at least one service provider SDP has

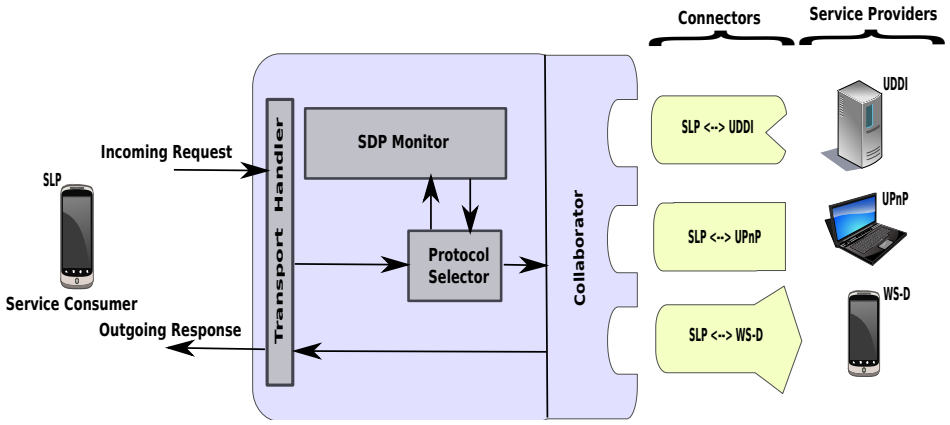


Figure 2: Interoperability System Architecture

the timeout less then or equal to the service consumer SDP timeout. Hence the *collaborator* knows that some responses will arrive after the service consumer SDP timeout. In order to counter this situation the *collaborator* maintains two timers, the initial timer *itimer* and the final timer $ftimer$ (see line 1– 2). The *itimer* timer makes sure that a message is sent without a message closing sequence before service consumer SDP timeouts, while the $ftimer$ timer makes sure that the message sequence is closed with successful aggregation. For the messages that arrived after the *itimer* timer timeout to be valid for aggregation, the *collaborator* will send the aggregated message from the buffer just before the *itimer* timer times out. This message will be sent without the message closing sequence and serves two purposes: Firstly, the service consumer SDP timer will wait for the remaining message to arrive with a renewed timer and therefore will not retransmit the same request again. Secondly, this will give enough time for the remaining messages that arrived after the *itimer* timer timeout to be aggregated. For instance, if a response from a service provider does not arrive before the $ftimer$ timeout the service provider is considered to be down, and the *collaborator* assumes all the responses are received and send the message closing sequence.

5. PROTOTYPE IMPLEMENTATION

We now present a prototype implementation of our interoperability system service discovery architecture. The core components of our prototype namely, *transport handler*, *protocol selector*, *SDP monitor* and the *collaborator* are implemented in Java to take advantage of cross platform portability. We use the z2z interoperable system [9] to implement *connectors* as it has been successfully demonstrated to provide good performance even on resource constrained devices. To access our prototype, we have deployed: (i) two service providers into a SNE, one compliant with UPnP and the other compliant with WS-Discovery, (ii) one SLP service consumer into a HNE, and (iii) our interoperable services into a gateway that interconnects the aforementioned HNE and SNE. In our testbed, the gateway is a Linux–based desktop machine with 3 network interfaces. Service providers are connected to both of them and the service consumer is connected to the gateway by the wireless interface. In order to demonstrate our proof-of-concept we have tested the 3 following use cases, as illustrated in Figure 3.

Algorithm 1 Message Aggregation Algorithm

1: set $itimer = Max\_Wait(Service\,consumer\,SDP)$
2: set $ftimer = Max\_Wait(Service\,providers\,SDP)$
3: **if** $Available\,Protocols \geq 1$ **then**
4: (i) Send Request(s).
5: (ii) Start $itimer$,$ftimer$
6: (iii) Wait for response(s).
7: **end if**
8: **while** $received\,Response$ **do**
9: **if** *Response* **then**
10: (a) Save Response to Buffer.
11: **end if**
12: **if** *itimer* **then**
13: **func_A()**
14: continue.
15: **end if**
16: **if** $packet\_size > packet\_size\_limit$ **then**
17: **func_B()**
18: continue.
19: **end if**
20: **if** $ftimer$ **then**
21: **func_C()**
22: break.
23: **end if**
24: **end while**
25: **func_A():**Send response message without closing sequence.
26: **func_B():**Send response message with overflow flag set.
27: **func_C():**Send response message with closing sequence.

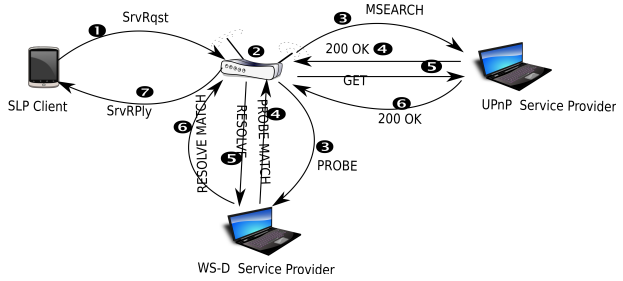


Figure 3: Testbed scenario

Use case 1 – The UPnP and WS-Discovery service providers provide an identical service named A. The client requests a service A (See Figure 3 ❶). The *SDP monitor* detects the presence of two SDPs in the network and with the help of the *collaborator*, two *connectors*, i.e. two z2z systems, are instantiated to translate incoming SLP messages to either UPnP or WS-Discovery messages, and UPnP and WS-Discovery replies to SLP (See Figure 3, ❸ ❹ ❺ ❻). At the end, the SLP client receives a single response advertising both services at different locations ❼.

Use case 2 – The UPnP service provider advertises a service named A, and the WS-Discovery service provider advertises a service named B. If the client requests the service A, two *connectors* are instantiated to perform a SLP to UPnP and a SLP to WS-Discovery translation. However, only one of the two service providers send a reply that is translated to the client.

Use case 3 – If one of the service providers goes down, the *SDP monitor* detects that one SDP is not anymore used, and consequently, its corresponding connectors is deallocated by the *collaborator*.

Our Prototype was tested using an in-house simulator to test the aforementioned use cases. For the simulations we manually changed the timeout for one the service to ensure that the reply could arrive after the initial timeout. For use case 1, we manually changed the timeout of service B and searched for the available services in the network. We then repeated the procedure by changing the timeout for service A and in both the cases we observed that our prototype successfully sent the reply as a single response to the client. For use case 2, we randonly searched for one of the two services and our prototype successful in detecting the requested service. while in use case 3, we randomly made one service provider to go down and *SDP monitor* successfully detected the correct SDP and deallocated the repective *connector*.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a scalable service discovery mechanism for large scale network using dynamic instantiation of component by our interoperability system that translate on–the–fly one SDP to another.

We present the design of our interoperability system and described a proof-of-concept prototype implementation. We tested our prototype using a in-house simulator and our prototype successfully discovered services in different networked environments using already existing service discovery protocols. This service discovery mechanism will be used in ALICANTE [2] to discovery services in different environment.

We are currently investigating how to extend our proposed architecture to support both service invocation and service delivery, so as to provide a full featured mechanism for service access in the context of Future Internet.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

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

[2] Alicante: media ecosystem depoloyment through ubiquitous content-aware network environments. *http://www.ict-alicante.eu/*.

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

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

[5] The salutation consortium : Salutation architecture specification. *http://systems.cs.colorado.edu/ grunwald/MobileComputing/Papers/Salutation/SA20E1D2.pdf*, 2011.

[6] R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, and R. Boutaba. Resource and service discovery in large-scale multi-domain networks. *Communications Surveys Tutorials, IEEE*, 9(4):2 –30, 2007.

[7] 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*.

[8] D. Benslimane, S. Dustdar, and A. Sheth. Services mashups: The new generation of web applications. *Internet Computing, IEEE*, 12(5):13–15, 2008.

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

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

[11] S. Cheshire and M. Krochmal. DNS-based service discovery. *http://tools.ietf.org/pdf/draft-cheshire-dnsext-dns-sd-10.txt*, 2011.

[12] S. Cheshire and M. Krochmal. Multicast dns. *http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-14*, 2011.

[13] L. Clement, A. Hately, C. Riegen, and T. Rogers. Universal Description, Discovery and Integration (UDDI). *http://uddi.xml.org/*.

[14] 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*, pages 1170–1187, 2003.

[15] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol (SLP).

http://www.ietf.org/rfc/rfc2608.txt.

[16] S. Ho and S. Kwok. The attraction of personalized service for users in mobile commerce: an empirical study. *ACM SIGecom Exchanges*, 3(4):10–18, 2002.

[17] W. Joy. JINI-Architecture. *http://www.jini.org/*.

[18] S. Khanna, J. Naor, and D. Raz. Control message aggregation in group communication protocols. *Automata, Languages and Programming*, pages 782–782, 2002.

[19] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.

[20] A. Mian, R. Baldoni, and R. Beraldi. A survey of service discovery protocols in multihop mobile ad hoc networks. *Pervasive Computing, IEEE*, 8(1):66–74, 2008.

[21] 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*.

[22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM, 2001.

[23] M. Raya, A. Aziz, and J. Hubaux. Efficient secure aggregation in vanets. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 67–75. ACM, 2006.

[24] S. C. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. In *USENIX Annual Technical Conference, General Track*, pages 127–140, 2004.

[25] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. 2001.

[26] H. Saleet and O. Basir. Location-based message aggregation in vehicular ad hoc networks. In *Globecom Workshops, 2007 IEEE*, pages 1–7. IEEE, 2007.

[27] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM, 2001.