ZigZag: A Middleware for Service Discovery in Future Internet

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

LaBRI – University of Bordeaux
Talence, France
{preston.rodrigues,david.bromberg,laurent.reveillere,
daniel.negru}@labri.fr

Abstract. Over the last few years, social networks, mobile devices and personalized services have been heavily responsible for a substantial increase in remote services available over Internet. Consequently, service consumers have to discover remote services anytime, anywhere across networks boundaries making thus service discovery, and their underlying Service Discovery Protocols (SDPs) more important than ever.

In this paper, we introduce ZigZag, a middleware to reuse and extend current SDP, designed for local networks, to discover available services across network boundaries as required in Future Internet. Our approach is based on protocol translation to enable service discovery irrespective of their underlying SDP. Further, we provide a thorough evaluation to validate our approach.

1 Introduction

The last decade has witnessed a tremendous growth of networked physical objects interconnected through Internet to form what is commonly named Internet of Things (IoT). More and more objects coming from different application domains such as mobile computing, handheld devices, embedded systems, sensors, and smart spaces are becoming connected to the Internet. However, the vast bulk of objects primarily connected to Internet have not been designed to interact seamlessly all together. 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.

In the new era of IoT, one key challenge is to enable interactions anytime anywhere with an unobtrusive connectivity among all interconnected objects across heterogeneous networks. One step towards that goal is to abstract networked physical objects, as either service consumers or providers, to only consider how services are discovered and accessed. However, service providers are not anymore solely confined to servers, and may potentially travels along with users across heterogeneous networks. Therefore, service discovery is a key bottleneck to the envision of a global infrastructure of networked objects as foreseen by Future Internet principles [1]. For instance, popular Service Discovery Protocols

K.M. Göschka and S. Haridi (Eds.): DAIS 2012, LNCS 7272, pp. 208–221, 2012.

(SDPs) such as SLP [11], WS-Discovery [4], and UPnP [17] do not allow a service provider to be discovered and accessed anymore if it moves from one network to another.

We have identified three main issues that current SDPs should resolve to provide service discovery in the large: (i) service identification, (ii) service interoperability, and (iii) 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 requests and responses 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. Finally, enabling the propagation of SDP requests to discover a remote service across many networks may produce a bulk of answers that a legacy client is not able to manage. Hence, different strategies may be explored and took in charge by the underlying infrastructure to provide to legacy clients the most adequate service according to non functional properties such as geographic locations, end-to-end delays, or available bandwidth. Therefore, aggregating SDP responses becomes an issue.

In this paper, we propose ZigZag, a middleware that enables SDPs initially designed for local area networks, to work across highly heterogeneous networks targeting the needs of the Future Internet and overcoming the aforementioned issues related to service discovery in Future Internet. Our approach is based on protocol translation to enable service discovery irrespectively of their underlying SDP. We have developed a prototype to support SDPs such as SLP, WS-Discovery, UPnP, UDDI. We have performed several experiments to assess our approach, demonstrating its benefits.

This paper makes the following contributions:

- We propose a middleware to instantiate and integrate any existing interoperability system to translate, on the fly, one SDP to another in the context of Future Internet.
- We introduce key building blocks to deploy in Future Internet. Their aim is to operate filtering and aggregation strategies to enables service consumers to find the most adequate remote services according to the criteria of their requests.
- We show the applicability of our solution by evaluating our approach using simulations.

The rest of this paper is organized as follows. Section 2 introduces our approach based on ZigZag. Section 3 presents in details the architecture of the ZigZag middleware. Section 4 describes the aggregation logic performed by the aggregator component of the ZigZag middleware. Section 5 presents the evaluation of our approach with the help of simulations. Section 6 discusses related works. Finally, Section 7 concludes and presents future work.

2 Service Discovery in Future Internet

One key cornerstone of Future Internet is to promote an architecture, split into two main planes, decoupling services from transport infrastructure. More precisely, one plane is dedicated to upper network layers to provide functions that control and manage service resources for service providers and consumers. The other plane is dedicated to lower network layers to provide functions that control and manage transport resources to carry out data exchanges among service providers and consumers across heterogeneous networks. The split of the architecture enables functions dedicated to services and the ones dedicated to transport to evolve separately and independently. As a consequence, Future Internet offers users unrestricted access to service providers outside their own network boundaries. However, this opportunity raises an issue related to service discovery. Indeed, legacy service providers rely on SDPs that have been initially designed for local area networks. Therefore, making these protocols scale to Future Internet requires a substantial effort. In addition, various protocols have been developed to cope with network characteristics and service providers needs. Thus, enabling service discovery in the context of Future Internet requires to manage the heterogeneity of various protocols, deployed in isolated local networks.

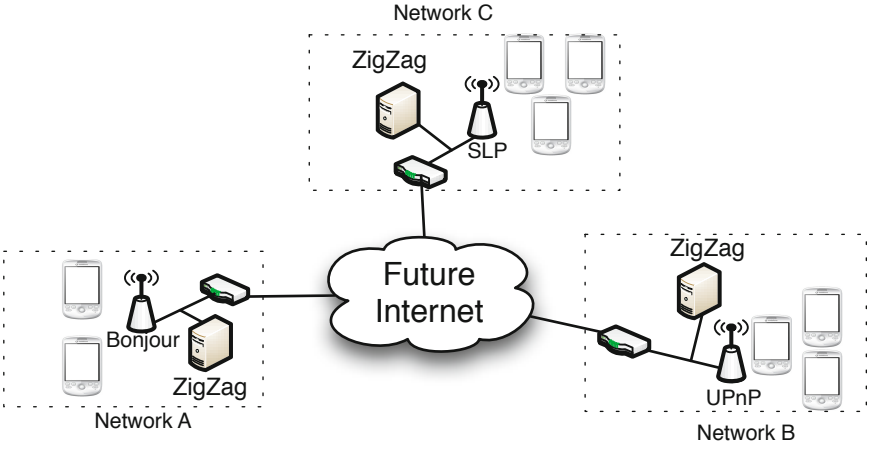

Fig. 1. ZigZag Approach

To overcome the issues of service discovery in the context of Future Internet, we introduce the ZigZag middleware. ZigZag has been designed to enrich upper layers of Future Internet while leveraging on the facilities available at the lower network layers. As illustrated in Figure 1, an instance of ZigZag is deployed in each isolated local area network. These ZigZag nodes monitor their own network and maintain a list of SDPs currently used. All ZigZag nodes communicate together thanks to the functions provided by the underlying Future Internet architecture. When a ZigZag node detects a request from a service consumer,

it forwards the request to remote ZigZag nodes. These nodes then instantiate appropriate connectors to translate requests and responses from the protocol used by the service consumer and the one used by the service provider. Once responses are received by the ZigZag node of the service requester, they are aggregated prior to being returned to the client.

3 The ZigZag Middleware

ZigZag aims to be deployed in many isolated local area networks to provide a service discovery in the large. To reach this aim, the architecture of ZigZag has been designed in a modular way to both integrate the state of the art results in service interoperability and our key contributions in service aggregations. ZigZag 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. As depicted in Figure 2, these components are plugged together to perform a cross network translation process that is able to translate one SDP to another according to service providers and consumers involved across heterogeneous networks. The core functionalities of each component are deeply explained below:

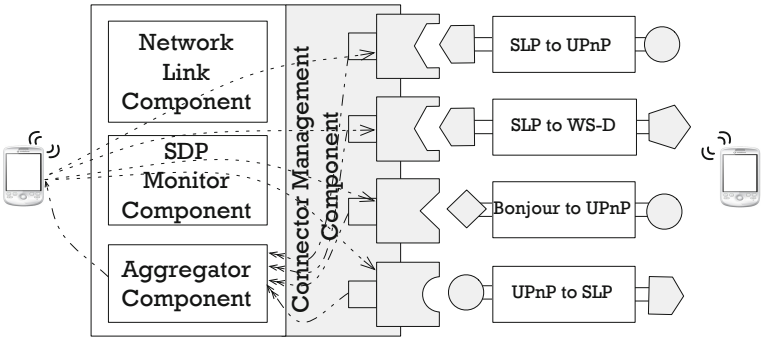

Fig. 2. ZigZag Middleware Architecture

SDP Monitor Component. The *SDP monitor* checks the availability of different SDPs in one local environment, as previously introduced by INDISS [5]. The SDP monitor is designed to keep track of SDPs currently used. It leverages on the fact that all SDPs use a multicast group address and a UDP/TCP port that must have been assigned by the Internet Assigned Numbers Authority (IANA). Both assigned ports and multicast group addresses are reserved and thus act as a permanent SDP identification tag. The SDP monitor is then able to discover a networked environment by passively listening to the well-known SDP

multicast group. More precisely, The SDP monitor learns the SDPs that are currently used from both services' multicast advertisements and clients' multicast service requests. Furthermore, service advertisements are cached locally and are mapped to a Universally Unique IDentifier (UUID) to 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. For instance, in Figure 2, four different connectors may be instantiated SLP-to-UPnP, SLP-to-WS-D, Bonjour-to-UPnP, UPnP-to-SLP according to the SDPs being used by either service providers and consumers. A connector is a third party component. Currently, the *Connectors Management Component* supports on the fly instantiation of one or more z2z gateways [6] that act as connectors. However, ZigZag is not tightly bound to z2z, and may rely on any other translator such as, for instance, Starlink [7]. Additionally, the Connectors Management Component collects statistics about SDPs being used to take in charge a fine grained life cycle of instantiated connectors. It may 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. The complexity of the Network Link Component implementation depends on the available functions supported by lower network layers. Currently, ZigZag supports ALICANTE [1] as the network infrastructure for the Future Internet, which provides adequate primitives (join, leave, update, send) to create and/or maintain a logical network among ZigZag nodes. Furthermore, ZigZag can also be deployed on different network infrastructure such as P2P overlay *via* the implementation of dedicated Network Link Components.

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 to then forwards it to the service requester. The aggregation logic of the Aggregator Component is described in more details in next section.

The message flow among different components is illustrated in Figure 2. A search request captured by ZigZag is span into as many search requests as there are different instances of compatible connectors. For Instance, according to the current configuration of the Connector Management Component described in Figure 2, if a SLP request is captured by the SDP Monitor Component, it is translated into both UPnP and WS-D requests. The different kinds of instantiated connectors depend on the information about compatible SDPs collected in

different isolated local networks by the SDP Monitor Component and exchanged by Network Link Component of each interconnected ZigZag middleware. The generated UPnP and WS-D requests are then forwarded either locally, and to remote isolated local network through the Network Link Component. In this example, many responses, which come from either locally and remote networks, are forwarded to the Aggregator Component to decide, according to its policy and the requester capabilities, to either aggregates them into one single response or select the one that matches best the criteria of the request, to finally generate a corresponding reply.

4 Aggregation

We now describe in more details the aggregation logic performed by the aggregator component of the ZigZag middleware, as introduced in Section 3.

4.1 Importance of Aggregation

Aggregation is defined as the process of accumulating data from multiple nodes to eliminate redundant transmission and provide value sensitive information to the requesting entity. In broadcast/multicast scenario 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 data aggregation is considered as the preferred solution. Response data 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 [13,19,20] have shown the importance of aggregation in different network environments. The authors in [18], presented an extensive survey on data aggregation techniques in the context of sensor networks. In a multi-protocol environment response messages may arrive after the source protocol signals a timeout as different protocols may have different timeouts due to their application design. In the following subsections we show the need of aggregation in a multi-protocol environment with the help of some use cases.

4.2 Use Cases

To illustrate the need of aggregation for service discovery in a multi-protocol environment we have identified 3 use cases based on different application requirements. In the following configurations, we consider a client $\mathcal{C}$ using SDP $\mathcal{P}_C$, and service providers $\mathcal{SP}_1, \mathcal{SP}_2, \mathcal{SP}_3$ and $\mathcal{SP}_4$ using SDPs $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ and $\mathcal{P}_4$ respectively. We assume that each service provider belongs to its own local area network and that a ZigZag node $\mathcal{ZZ}_i$ is deployed in each network.

Information Craving. In this scenario, the application running on client $\mathcal{C}$ does not have a stringent timeout constraint. Indeed, the application can wait until all possible responses from available service providers are received. The timeout of the client is greater than each timeout of $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ and $\mathcal{P}_4$. Request from $\mathcal{C}$ is sent to all ZigZag nodes. Received responses are then aggregated and a unique response is sent back to $\mathcal{C}$.

Time Bound. In the second scenario, the client application requires replies within a very short period of time. Indeed, the client would signal a timeout if no response is received before its timer expires. Once the request from $\mathcal{C}$ is forwarded to all ZigZag nodes, a timer is started so that an aggregated response can be sent on time, before the expiration of the timer of the client. Therefore, responses that arrive too late are discard.

Best of Both Worlds. The third and last scenario is a mix of the previous ones. In this scenario, the client $\mathcal{C}$ tolerates one timeout expiration and one request re-submission. The request from the client is forwarded to all ZigZag nodes. Responses are cached once they are received and requests are submitted again if one timer expires. Regardless of the responses received, all responses are aggregated and one response is sent back to the client before the second expiration of its timer.

4.3 Aggregation Policy

The aggregation policy describes how to aggregate value sensitive data from different protocols having variable message timeouts until a response forwarding condition $\mathcal{Fc}$ is satisfied. A policy can consist of selecting the response that matches the best a specific criteria or aggregating several replies from different service providers to provide to the client the best valuable response. However, the amount of time a ZigZag node can wait to satisfy $\mathcal{Fc}$ is bound by the timeouts of the different protocols involved, the time required for translating messages from one protocol to another, and the time for exchanging messages from ZigZag nodes. Based on these information, a ZigZag node can decide to return a provisional response to the client to avoid a timeout expiration. Aggregation policies are implemented using building block functions provided by the ZigZag middleware. Our current implementation of ZigZag supports the Python and C programming languages to define policies. We have defined the aggregation policy for each scenario previously introduced. We present in the next section an assessment of our approach.

5 Evaluation

We have developed a prototype implementation of ZigZag. Our current implementation relies on z2z [6] to dynamically instantiate gateways for protocol translation and Alicante [1] to connect ZigZag nodes to each others. Alicante is an

on-going ICT project funded by the European Commission that propose a new architecture to support the deployment of a networked *Media Ecosystem*. To this end, it introduces two novel virtual layers on top of the traditional Network layer, *i.e.* a Content-Aware Network layer (CAN) for network packet processing and a Home-Box layer for content delivery. These virtual layers enables ALICANTE architecture to be deployed over several geographical regions forming large scale networks. ZigZag is being integrated in ALICANTE and we expect to perform real world experiments in that context. Indeed, a distributed multi-domain networking environment composed of pilot islands located in different countries will be delivered during the project.

Before performing real world experiments of ZigZag, we have setup a simulation to assess how much ZigZag can both reduce the number of messages that flow through the network, and provide value sensitive information to the requesting entity. In the remainder of this section, we explain the simulation parameters and discuss the results we have obtained.

5.1 Simulation Setup

To simulate various clients, service providers, network topologies and protocols prior to large scale deployment, we have performed a simulation based on SimPy [15], a network simulator written in Python. To provide the most realistic result, and to outline an accurate evaluation of our prototype, we include in our simulation an adequate model of the *Internet delay space*, which influence inherently the ZigZag performance. In particular, we leverage on a real sample of the Internet delay space among 3,997 edge networks [21] to build our overlay. Correspondingly, we rely on a 3997x3997 delay space matrix that gives all pairs set of static round-trip propagation delays among nodes of our overlay network. Service providers based on either SLP, UPnP, WS-D, or Bonjour are then randomly distributed uniformly over the matrix. Each node hosts only one type of service.

We run our experiments 50 times with three different client instances: $\mathcal{C}_1$ that uses UPnP, $\mathcal{C}_2$ that uses Bonjour, and $\mathcal{C}_3$ that uses SLP randomly located in one node of the overlay. At each run, clients are located in a different node. A request from a client is generated according to the Poisson process with a rate of 5 requests per seconds for a simulation period of 200 seconds. The processing time P_{time} of a service provider, to send responses upon the reception of a request, is $P_{time} = k \times (round\_trip\_delay/2)$ where k is a factor randomly chosen from 2 to 3.6 according to SDPs specification [11,16]. An infinite response time means that the service provider is overloaded and that the request has been dropped. The time required by a ZigZag node to translate a message from one protocol to another depends of protocols used by both the client and the targeted service provider. Table 1 shows the different possible translation time. To define these time values, we computed the average time consumed by z2z connectors [6] to perform the translation. In the remaining, we always assume that a ZigZag node is deployed in the local area network of the requesting clients, thus the round-trip propagation delay among requesting clients and its closest

ZigZag node corresponds to the round-trip delay of a 100 Mb/s LAN network. The forwarding of SDP requests from clients to one or more adequate remote service providers across the overlay is provided by a service provider selection algorithm that should redirects SDP requests from clients to an appropriate remote service provider, based on factors such as the client location, network conditions, processing load, service search criteria and other parameters dedicated to ALICANTE [1] network substrate. However, such model for ALICANTE is not yet available. Thus, we use a selection algorithm that picks up required service providers randomly in the delay space matrix to get their round trip delay. A more accurate algorithm is planed to be used and will give more efficient simulation results.

Table 1. Average translation time (in seconds)

	SLP	UPnP	WS-D	Bonjour
SLP	0	0.78	0.84	0.59
UPnP	0.78	0	0.65	0.89
WS-D	0.84	0.65	0	0.90
Bonjour	0.59	0.89	0.90	0

We now present the results of our simulation using the uses cases described in Section 4.2.

5.2 Simulation Results

The median results of our simulations are shown below. In the simulation results, the x coordinate indicate the number of services that have been reached by clients during the simulation whereas the y coordinate gives the number of generated messages.

Information Craving. Figure 3 shows the results of our simulation for the *Information craving* scenario. In this scenario, as the timeout of one client is greater than each timeout of SLP, UPnP and Bonjour, only one client is required to generate requests, and the simulation result corresponds to the best case as the client timeout is high. 82% of the received responses are aggregated, with at least 2 or more responses per message. About 16% of non aggregated messages are received. About 2% of the messages are lost due to errors in the network or replies that did not arrive on time.

Time Bound. Figure 4 shows the results of our simulation for the *Time bound* scenario. In this scenario, three different client has been used, one for each SDP, i.e. SLP, UPnP and Bonjour. Each client uses therefore a different timeout. In average, 62% of received responses are aggregated with at least 2 or more

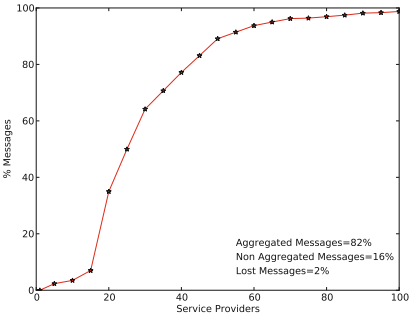

Fig. 3. Scenario 1 – Information craving

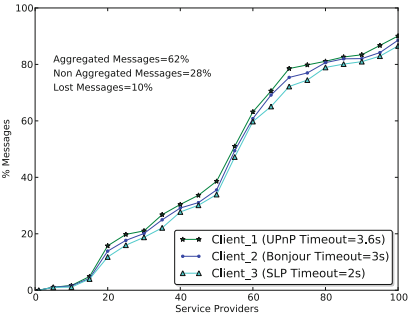

Fig. 4. Scenario 2 – Time bound

responses per message. Further, about 28% of non aggregated messages are received and 10% of the messages are lost due to errors in the network or replies that did not arrive on time. The increase of both non aggregated messages, and lost messages comes from the fact that some replies arrived after the client has signaled a timeout, and are thus ignored. Compared to the previous results, the simulation results obtained in this scenario corresponds to the worst case.

Best of Both Worlds. Figure 5 shows the results of our simulation for the *Best of both worlds* scenario. In this scenario, in average, clients get 70% of aggregated responses with at least 2 or more responses per message and about 23% of non aggregated messages. About 7% of messages are lost due to errors in the network or replies that did not arrive on time. The successful increase of aggregated responses comes from the fact that clients tolerates one timeout expiration plus one request re-submission. It means that ZigZag nodes are caching all responses in order to aggregate them to reduce the number of requests sent back to the client.

Table 2 gives a summary of our simulation results. Our experiments have demonstrated that ZigZag enables between 57% to 80% of received messages to be the aggregation of at least two service provider replies. This high rate of aggregated messages implies a significant reduction in the number of messages exchanged through the network and more valuable answers to the client. Figure 6 shows a comparison of all the three use cases. In addition, various policies can be easily implemented and deployed using ZigZag to cope with user expectations and network constraints.

Table 2. Simulation summary

	Average number of messages received by a client		
	Aggregated	Non-aggregated	Lost/Dropped
Information craving	82 %	16 %	2 %
Time bound	62 %	28 %	10 %
Best of both worlds	70 %	23 %	7 %

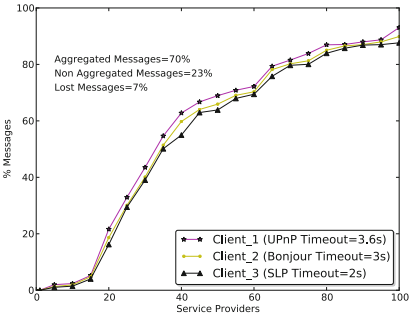

Fig. 5. Scenario 3 – Best of both worlds

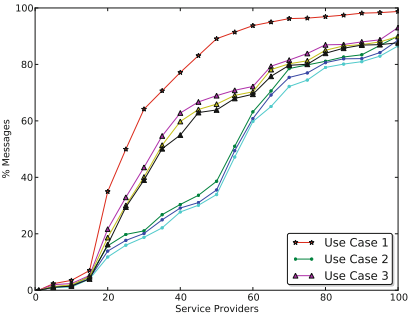

Fig. 6. Use Case Comparison

6 Related Work

Service discovery is a mechanism that enables service consumers to find remote services without having any previous knowledge of either their locations or their characteristics. There are many Service Discovery Protocols (SDPs) available each having a specific discovery technique to discover services in the network. SDPs rely on either centralized, decentralized or hybrid service directories to store their service information. An extensive literature survey [3,14] 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 [1] [11], WS-Discovery [2] [4] and UPnP [17], 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 [12], Salutation [2] and Bonjour [8,9]; both service consumers and providers need to multicast requests to find an available service directory. On completion of the above process, service providers are able to register their services using a

[1] SLP can also be configured to rely on a central service directory (Directory Agent).
[2] WS-D can dynamically configures itself to rely on a central service directory (Discovery proxy).

unicast request, and service consumers can now make a unicast request to search for available services.

SDP Interoperability. 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 [10], INDISS [5], z2z [6] and Starlink [7]. ReMMoc is a reflective middleware that provides a specific API to hide to applications the underlaying SDPs currently used in the local network environment. However, ReMMoC requires from 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. However, extending INDISS to support new protocols is a challenging task as it requires both a deep knowledge of the protocols involved, and also a substantial understanding of low-level network programming. 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.

However, all the aforementioned solutions do not address SDPs interoperability crossing local network boundaries and are hence unusable in the context of Future Internet. To address this issue, we provide ZigZag that leverages on the state of art to provide a large scale service discovery.

7 Conclusion and Future Work

In this paper, we propose ZigZag, a middleware that enables SDPs initially designed for local area networks, to work across highly heterogeneous networks targeting the needs of Future Internet. Our approach is based on protocol translation to enable service discovery irrespectively of their underlying SDP. We have developed a prototype implementation of ZigZag that instantiates on the fly z2z as connectors and supports Alicante as the network infrastructure for the Future Internet.

We have performed several experiments using service discovery protocols such as SLP, UPnP, WS-Discovery or Bonjour. We have performed a simulation to assess our approach, demonstrating its benefits. Our experiments have demonstrated that ZigZag enables between 57% to 80% of messages to be aggregated, thus reducing network usage and increasing valuable responses for users.

We are currently investigating a number of research avenues. We plan to extend the ZigZag middleware to support both service invocation and service

delivery, taking full architectural advantage of Future Internet. We are also exploring the definition of a domain-specific language to describe a user-specific aggregation policy. This language approach should raise the level of abstractions, enabling users to specify themselves how they would like responses to their requests to be aggregated. This approach should also enable verification and optimization to be performed by ZigZag, to take into account non-functional properties of each ZigZag node.

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. Alicante: media ecosystem deployment through ubiquitous content-aware network environments, http://www.ict-alicante.eu/
2. The Salutation Consortium: Salutation Architecture Specification, http://systems.cs.colorado.edu/ grunwald/MobileComputing/Papers/ Salutation/SA20E1D2.pdf
3. Ahmed, R., Limam, N., Xiao, J., Iraqi, Y., Boutaba, R.: Resource and service discovery in large-scale multi-domain networks. IEEE Communications Surveys Tutorials 9(4), 2–30 (2007)
4. Beatty, J., Kakivaya, G., Kemp, D., Kuehnel, T.: Web Services Dynamic Discovery (WS-Discovery), http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec .html , http://docs.oasis-open.org/ws-dd/discovery/2009/01
5. Bromberg, Y.-D., Issarny, V.: INDISS: Interoperable Discovery System for Networked Services. In: Alonso, G. (ed.) Middleware 2005. LNCS, vol. 3790, pp. 164–183. Springer, Heidelberg (2005)
6. Bromberg, Y.-D., Réveillère, L., Lawall, J.L., Muller, G.: Automatic Generation of Network Protocol Gateways. In: Bacon, J.M., Cooper, B.F. (eds.) Middleware 2009. LNCS, vol. 5896, pp. 21–41. Springer, Heidelberg (2009)
7. Bromberg, Y.D., Grace, P., Réveillère, L.: Starlink: Runtime interoperability between heterogeneous middleware protocols. In: Proceedings of the 2011 31st International Conference on Distributed Computing Systems, ICDCS 2011, pp. 446–455. IEEE Computer Society, Washington, DC (2011), http://dx.doi.org/10.1109/ICDCS.2011.65
8. Cheshire, S., Krochmal, M.: DNS-based service discovery (2011), http://tools.ietf.org/pdf/draft-cheshire-dnsext-dns-sd-10.txt
9. Cheshire, S., Krochmal, M.: Multicast dns (2011), http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-14
10. Grace, P., Blair, G.S., Samuel, S.C.: ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS/DOA/ODBASE 2003. LNCS, vol. 2888, pp. 1170–1187. Springer, Heidelberg (2003)
11. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol (SLP), http://www.ietf.org/rfc/rfc2608.txt

12. Joy, W.: JINI-Architecture, http://www.jini.org/, http://www.jini.org/wiki/Jini_Architecture_Specification
13. Khanna, S., Naor, J., Raz, D.: Control Message Aggregation in Group Communication Protocols. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 135–146. Springer, Heidelberg (2002)
14. Mian, A., Baldoni, R., Beraldi, R.: A survey of service discovery protocols in multihop mobile ad hoc networks. IEEE Pervasive Computing 8(1), 66–74 (2008)
15. Muller, K.: SimPy Simulator, http://simpy.sourceforge.net/
16. Plug, U.: Play (UPnP). Internet (April 17, 2007), http://www.upnp.org
17. Presser, A., Farrell, L., Kemp, D., Lupton, W.: Upnp device architecture 1.1, http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf
18. Rajagopalan, R., Varshney, P.: Data-aggregation techniques in sensor networks: a survey. IEEE Communications Surveys & Tutorials 8(4), 48–63 (2006)
19. Raya, M., Aziz, A., Hubaux, J.: Efficient secure aggregation in vanets. In: Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks, pp. 67–75. ACM (2006)
20. Saleet, H., Basir, O.: Location-based message aggregation in vehicular ad hoc networks. In: IEEE Globecom Workshops, pp. 1–7. IEEE (2007)
21. Zhang, B., Ng, T.S.E., Nandi, A., Riedi, R., Druschel, P., Wang, G.: Measurement based analysis, modeling, and synthesis of the internet delay space. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC 2006, pp. 85–98. ACM (2006)