Optimized Contextual Discovery of Web Services for Devices

Nicolas BUSSIÈRE[1] Daniel CHEUNG-FOO-WO[1,2] Vincent HOURDIN[1,3]
Stéphane LAVIROTTE[1,4] Michel RIVEILL[1] Jean-Yves TIGLI[1]
[1] Laboratoire I3S, Université de Nice - Sophia Antipolis - CNRS
[2] CSTB 290, route des Lucioles, BP209 06904 Sophia Antipolis
[3] Preceptel Buropolis 1240, Route des Dolines 06560 Sophia Antipolis cedex
[4] IUFM Célestin Freinet - Académie de Nice 89, Avenue George V 06046 Nice Cedex 1
{bussiere, cheung, hourdin, lavirott, riveill, tigli}@polytech.unice.fr

Abstract

Due to more and more mobile computers moving among smart and communicating devices in our everyday life, we observe the emergence of new constraints in software design. Indeed, device heterogeneity, dynamic software variation, and frequent mobile device apparition/disappearance make software applications compulsorily adapt to their context. In this paper, we will present an enhancement of ambient computing discovery mechanisms adding context handling capabilities to Web Services for Devices. As a matter of fact, we define contextual parameters for broadcast requests so that only devices in the "selected context" reply, which aim at reducing the overall number of exchanged messages on the network.

1 Introduction

With the multiplication of mobile computing devices and communicating devices, we observe the emergence of new constraints in software infrastructures. We observe as well the emergence of a computer science called "**Ubiquitous Computing**" by Mark Weiser [11] also called **Ambient** or **Omnipresent Computing** based on the following statement: "*Silicon-based information technology is far from having become part of the environment*". Technical difficulties being overcome day after day, numerous physical entities of our environment progressively acquire a computing existence and new communication capabilities; more concretely, they participate in new software applications [10].

1.1 Service-oriented architecture for devices

To fulfill reusability and adaptability needs, **Service-Oriented Architectures** (**SOA**) have been defined. Their principle is to describe services and their interactions [6]. The SOA approach presents interesting benefits for device management. We now present the combination of SOA and devices.

We call input/output devices, or simply **devices**, equipments interacting with the physical environment. Devices can be sensors, actuators or equipments of various nature which can be found in new interaction supports for ambient computing. Devices are basically similar to services as they both provide a functionality, a means to access it, and also because they are loosely coupled. However, **physical constraints** – also called **resource dependencies** – make them different. A device can appear or vanish at any time. Its visibility can be considered as a contextual information.

Whereas projects such as SIRENA [3] based on a **SOA for devices** outline equipment integration facility, reusability, general system extensibility at runtime, and interoperability between services, we focus particularly on **Web Services for Devices** (**WSDs**). We will group under this denomination approaches aiming to council **Web Service** principles and **device** utilization (physical resource dependent by nature). The best-known approaches dealing with WSD are UPnP [1] and more recently DPWS [2].

Those two previous WSD implementations are not interoperable. This is a good example of the **heterogeneity** problem of Ambient Computing at the protocol level. [8] presents a solution which deals with such limitations of protocol interoperability by providing a bridge mechanism. We can distinguish three kinds of heterogeneity: (1) heterogeneity of networks and protocols introduced before, (2) heterogeneity of platforms, and (3) heterogeneity of services. Our work is essentially focused on the latter. Namely, by adding an abstraction layer on top of devices, Services for Devices are allowed to interact by pro-

[1] http://upnp.org/resources/documents.asp
[2] http://schemas.xmlsoap.org/ws/2006/02/devprof

viding standard communication protocols and functionality descriptions. **Web Services for Devices** specially aim at permitting **interoperability**, since they do not rely on any specific programming language nor hardware architecture.

Service for Devices also undergoes constraints related to devices' **resource dependencies**: frequent disconnections, memory limitation, narrow network bandwidth, limited power, processing capacities, etc. Therefore, the description of Service for Device must include these limitations to inform of specific constraints associated to the provided service.

To go further, more specific works on the description of ontological devices and services such as [2] are thus necessary [1] to give a complete Service for Device description.

Devices being most often connected to the real environment of applications, associated services need to offer mechanisms which take into account applications' **reactivity** to environmental variations. However, this is not without consequences on the WSD communication protocols. WSDs thus define communication protocols using events (subscription, notification) in an asynchronous execution context. For this purpose, UPnP uses the GENA protocol[3], whereas DPWS holds on WS-Eventing[4].

Besides, the **location** concept is largely used to determine the availability of services for devices. In software usage, location is implicitly linked to user's proximity inside his environment [7].

Centralized service directories are difficult to keep up to date, because applications undergo frequent disconnections of devices, which leads to communication overheads while keeping coherence between pieces of information stored in a directory. Moreover, evaluating the proximity between a device and a user is not easy to compute. UDDI (Universal Description, Description and Integration of Web Services)[5] is the main standard for Web Services' publication and discovery. Web Services export their utilization interface through the Web Service Description Language (WSDL)[6] , which ensures the independence from the technical platform used by the service provider.

Adopted solutions in WSD hold on local and **distributed discovery** mechanisms between service producers and service consumers. It is the case for UPnP and DPWS, with respectively SSDP protocol and WS-Discovery protocol[7].

These protocols use multicast UDP messages to discover servers on the local network. There are actually two different processes for discovery: search and advertisement. Search is an active request made by clients to discover servers on the network. An advertisement is made by the servers when they arrive on the network, then periodically, and finally, when they quit. The location concept is therefore linked to message routing and associated to the same local network membership.

However, it seems more natural to associate service discovery to context. Indeed, the interest of a service for device in the user's environment is not only linked to a simple geographic criterion. We can for example associate the service availability and its discovery to some authorized time intervals and more generally to contextual exploitation conditions [5]. Some works outline the context introduction in service discovery mechanisms [4].

We can cite a third type of discovery management, the local directory, which gathers centralized service directory and distributed discovery. This directory listens to local network announcements and quit messages, and keeps up to date a service list. This means that it is transparent for services, unlike a pure centralized directory. The best example of local directory is the DPWS' Discovery Proxy[7]. It is an optional entity in WS-Discovery used to minimize broadcast or multicast discovery messages, which can be expensive for wireless networks.

1.2 Context model

The term "context" has been used to describe various concepts such as positions (characterizing the physical context), resources (computing context) and human interactions (social context). Gathering all these different concepts often leads to ambiguities. Each community having a different definition for context, finding a common meta-definition is quite a hard job. We rely on the formal representation of context of [7, 5] which introduces a different approach to context-awareness by conveying asymmetric relations between devices.

Numerous works on context consider that there exists a symmetric relation between contextual entities, but this reciprocity is not always true. Indeed, having an A entity in our context does not mean that we are in the context of A (loving someone does not necessarily means to be loved). We can thus distinguish devices which are in our context (**endo-context**) and devices for which we are in their context (**exo-context**). A device can select other devices which are in its context (**endo-selection**) or devices for which it is in their context (**exo-selection**). When both relations are verified (both devices are in the other's context), the selection mechanism is said to be **bilateral**. We refer the interested reader to [5] for more details on these different selection mechanisms.

In this **contextual asymmetric** world, each entity needs a function to compute contextual membership; let us call it the ϕ function. Entity e_1's ϕ function is used to check if an entity e_2 is in its context; this is done by passing contextual

[3]http://quimby.gnus.org/internet-drafts/draft-cohen-gena-p-base-01.txt
[4]http://www.w3.org/Submission/WS-Eventing
[5]http://www.uddi.org/specification.html
[6]http://www.w3.org/TR/wsdl
[7]http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf

data of e_1 and e_2 to ϕ. The phi function can compute a distance, which is a symmetric function, but for most of other cases this is an asymmetric cost function, used for an exo-selection of the context (i.e. if e_2 is in the context of e_1).

We consider devices to be the basic entities of which we will study the context.

2 Contribution: contextual discovery of web services for devices

As we have seen with SOA and particularly WSD, services can be discovered and added to an application during its execution. However, searching for any devices on a network may retrieve a large amount of unneeded data flows, resulting in resource wastes (network bandwidth, energy, time).

In the UPnP case, search messages may use a filtering mechanism to search for a device with a specific unique identifier, for a device of a given type, or for a device providing a given type of service. This filtering mechanism is not efficient enough to be used for a more generic contextual filtering as all devices of a given type may not be in the user's or application's context. Since devices are not known beforehand, the unique identifier filter is useless as it can only be applied on previously found devices.

DPWS provides two kinds of service filtering: on their type and on their scope. The type filter is similar to the filtering mechanism described earlier for UPnP. The scope filter is much more promising. Scopes can be used to organize services into logical groups, for example one for a department, or for a specific floor in a building. They can be specified with a LDAP URL syntax, thus the possibilities are wide.

The solution we are exposing in this paper relies on the ability to search devices for which the application is in their context (exo-selection). This is a non-trivial and efficient implementation from the formal description described in [5] which also tries to minimize the overhead communication costs due to context selection. In an active discovery (flooding), devices must have a ϕ function, used to compute the contextual memberships. In a passive discovery (advertisement), they must be able to send their ϕ to the interested entities. As we have seen, the function ϕ uses the context values of both devices (searching device and potential provider device) to compute the membership. Thus, discovered devices must have a ϕ function and also context information on themselves.

We can distinguish two kinds of devices: (1) devices that produce information requested by the application (let us call them **D** devices) and (2) devices informing about the context (we will refer to them as **C** devices). **D** devices may refer to different **C** devices to define values of their context information. A **C** device can finally be shared among several **D** devices. If a **D** device is not linked to a **C** device, it does not depend on context. Although **C** device can exists without being used by any **D** devices, we choose not to model this situation which does not present a great interest as it would mean that there is no application. Depending on the application, a same device can be viewed either as a **C** or a **D** device. For example, for greenhouse applications, temperature can be viewed as non-contextual information although we can perform a temperature-oriented search when looking for an available fresh classroom.

Enabling WSD to achieve contextual search and discovery means, with the above notation, that **D** devices will be discovered only if their associated **C** devices match the context selection. There are at least four different ways to do this, and we will study them in the next section. The whole idea is to add context information to discovery messages or search responses of WSD. The traditional way would have been to request the context instance from every found device, increasing the workload and unneeded network traffic.

Making **D** devices aware of their context is not an easy work. Our solution is to aggregate **C** and **D** devices together into a single new WSD, by means of an orchestration of WSD [9]. We instantiate, in our event-based component framework, proxy components for WSD (there for **C** and **D** devices) in a container, and manipulate them like software components. The container is able to export its own assembly of components as a WSD, which additionally makes the model re-entrant. This way, search requests and discovery messages are processed by the upper level aggregated WSD (the component assembly), which internally makes needed requests to the concerned **C** devices.

Context information can be added to UPnP messages. For instance, a UPnP search request is a HTTP header defined by fields such as response delay and search target, without a HTTP body. We can add context information in the body of the request, most of standard UPnP stacks are not disturbed if headers are still the original ones. For context-aware UPnP servers, a modification in the SSDP parser has to be done to check context membership before replying to the discovery. There is no need for such modification with DPWS, since scopes can be user-defined, they can be used to represent an URL-expressible context.

3 Contextual search model

In this section, we will explain in detail how an active contextual discovery of WSD can be done. Each presented solution lets us find devices in a given context, on a local network. Thanks to the active search messages' modeling, we will compare the costs of each approach in terms of the number of exchanged messages.

3.1 Case A: basic application

The obvious and simplest way to perform a context-aware search of Web Services for Devices is to do a non context-aware search on the network for servers with a multicast message in a first time, and then to ask every found device for its context.

We note N the total number of devices, which is the sum of P, the number of **D** devices that produce information requested by the application and K, the number of contextual **C** devices that gives information about the context of **D** devices. We note Cost_d the cost of the discovery and Cost_c, the cost of the communication used to determine if the context matches or not (in most cases, a SOAP request).

We assume that the contextual **C** device transmits the complete contextual information in a unique request packet.

We have the following formulas :

$$\text{Cost}_\text{d}=\gamma+\alpha N$$

$$\text{Cost}_\text{c}=(\beta_1+\beta_2)K$$

where γ stands for the initial discovery messages, α the number of messages in response and β_k represents variables depending on the communication infrastructure. Here, β_1 is the number of packets needed for the context request, and β_2 for the response. All N devices respond to the search request, but the context request is made only to the K **C** devices, which gives contextual information about **D** devices.

3.2 Case B: aggregation of 'D ' devices only

In this second case, we aggregate **D** devices together. When searching for devices, we will only discover these aggregates, but we still have to request the contextual information to the **C** devices.

The number N now stands for the number of devices' aggregates. We note p_i the number of **D** devices and k_i the number of **C** devices that are contained in the i^{th} aggregate.

We consider two sub-cases: **D** services are aggregated into a new device but not their interfaces, such as there will be a service per former **D** device (case 1), and services are aggregated and their services are merged into a single new interface (case 2).

Case $\mathbf{B}_1$: **D** services are aggregated

$$\begin{aligned}\text{Cost}_\text{d} &= \gamma+\alpha\sum_{i=1}^{N}p_i\\ \text{Cost}_\text{c} &\leq (\beta_1+\beta_2)\sum_{i=1}^{N}p_ik_i\end{aligned}$$

All **D** devices in aggregates respond to the search message, and context requests are done to all **C** devices associated to a **D** device, for each **D** device, in all aggregates. As a **C** device may be associated to several **D** devices for their contextual information, Cost_c is not a strict equality, and may be lower.

Case $\mathbf{B}_2$: services are merged

$$\text{Cost}_\text{d}=\gamma+\alpha N$$

$$\text{Cost}_\text{c}=(\beta_1+\beta_2)\sum_{i=1}^{N}k_i$$

In that case, services are merged, and we only have the aggregated **D** devices which respond to the search request. Then, for the context request, **C** devices are associated to the new **D** device, the request will be done on each of them, but only once per aggregated device.

3.3 Case B′: aggregation of contextual C devices and D devices

This third case is similar to the previous one (B), except that **C** devices are now in aggregates of devices. This implies that if an aggregate of device contains more than one **C** device, context requests will be handled internally and the external cost will be as if there were only one request to fetch all context values for all **C** devices in each aggregate.

Case $\mathbf{B}'_1$: **D** services are aggregated

$$\begin{aligned}\text{Cost}_\text{d} &= \gamma+\alpha\sum_{i=1}^{N}p_i\\ \text{Cost}_\text{c} &\leq (\beta_1+\beta_2)\sum_{i=1}^{N}p_i\end{aligned}$$

D devices in aggregates respond to the search request. Context requests are made for each of those in every aggregate.

Case $\mathbf{B}'_2$: services are merged

$$\text{Cost}_\text{d}=\gamma+\alpha N$$

$$\text{Cost}_\text{c}=(\beta_1+\beta_2)N$$

Now, only aggregated devices can be discovered, and context requests can be done directly to them for all nested devices.

3.4 Case C: our contribution: deporting the contextual detection to the device

As we have seen in the section 2, our solution is revealed efficient as it enables devices to communicate their context information in search replies. Therefore, context requests are no longer needed.

We note r the ratio representing the number of devices belonging to the context of the application ($0 \leq r \leq 1$). In this case, the cost of the communication is null. We have then respectively the formulas for the two cases:

$$\text{Cost}_{\text{d}}=\gamma' + \alpha r N$$

$$\text{Cost}_{\text{c}}=0$$

3.5 Summary: gain in message number

The important thing we can depict from these equations is the decrease of the number of messages in the C case. We can analyze the ratio corresponding to this decrease for different approaches. For the case B'_2 and C, we have :

$$\text{ratio} = \frac{\text{Cost}_{\text{C}}}{\text{Cost}_{\text{B}'_2}} = \frac{\gamma' + \alpha r N}{\gamma + \alpha N + (\beta_1 + \beta_2) N}$$

When N, $\sum^{i} p_i$ and $\sum^{i} k_i$ are big enough, this tends to

$$\text{ratio} \longrightarrow r \frac{\alpha}{\alpha + \beta_1 + \beta_2}$$

4 Experiment and validation

In this section, we will study a real-world application case. Let us take a typical study week in a last year university cycle. We have there 67 students following 6 among 19 lectures, given by 13 teachers. Teachings spread over half-days, and often consist of a course followed by a directed tutorial class or a practical work by halves; so 2 hours will be our basic time granularity.

All teaching rooms, the center hall, and teachers' offices are equipped with interactive devices, which can be discovered as WSD. All those devices constitute our non-moving devices (that we will call further "fixed devices") network. We also have in every location context capture devices giving context information for all the fixed devices.

This sample would not be interesting if students were not carrying moving devices. Actually, they are equipped with some mobile devices, like a cell phone or a PDA, connected to the main network as well. Some tracking system, or mobile device localization system is coupled to mobile devices, allowing an application to know its context by itself.

	Light	**Shutter**	**Display**	**Loud Speaker**	**Large Display**	**Video-projector**	**Air-conditioning**	**Number of room**	**Devices per room**	**Total**
Course	2	2	2	1	0	1	1	3	9	27
Directed Work	2	1	1	0	0	0	1	2	5	10
Practical Work	2	1	1	0	0	0	0	3	4	12
Hall	6	4	1	0	1	0	0	1	12	12
Office	1	1	1	0	0	0	1	14	4	56
									34	**151**

Table 1. Classrooms and fixed devices

The typical application which can be set up with this environment is the following: a teacher coming in a classroom to give his course would like to connect to the wireless video-projector to display his documents and to the loudspeakers to amplify his voice. If he discovers devices without any filter, he will find all devices in the building. If he specifies a type, he will discover all video-projector devices in the building. If he specifies a type, and a scope, if we consider we are in a DPWS environment and scopes are defined with classrooms, he will find the video-projector he needs. But if students proceed to the same discovery, they will be able to find it too, what we surely don't want to happen in a general case. Therefore, there is still a need for a contextual evaluation. In that case students will not be able to discover the video-projector because they are not in its context, contrarily to the teacher.

The data we are taking for the following quantitative experiment include devices worn by the students. A teacher can proceed to a search on them, for example to send them examination questions or instructions for a practical work.

Table 1 describes which devices can be found in every kind of room. We count 151 fixed devices in our network.

Starting from the courses distribution in time, and the number of students who have registered to lectures, we calculated the ratio r for each time interval and each room. The maximum value found for r is 0.28, which means that every time, discovered devices are nearly a quarter of the whole set at its maximum, with an average value of 0.17 and a standard deviation of 0.05. Network traffic will be lowered as much, as well as the ease of use of the system.

5 Conclusion

In this paper, we presented contextual discovery based on Service-Oriented Architecture for Devices, well suited to manage heterogeneity, distribution, and dynamicity in an Ubiquitous Computing environment. The specificity of Web Services for Devices lies in its adequacy to manage

interoperability, reactivity and location of services for devices. But with a wide range of devices, the problem of search and discovery of available services is a challenge, due to the number of messages. To solve this problem, we presented our definition of context, especially the asymmetry of this notion (endo and exo selection). Then we applied this to provide a contextual discovery of devices, enabling to have better performances in term of communication messages to discover such contextual devices. Finally, we presented an example applying these researches in a real environment, to highlight the gain obtained with contextual discovery of mobile devices.

In the university building sample, we used an exo context selection mechanism. In future research, we can consider endo selection mechanism performance enhancement by deporting the contextual area computing (ϕ function) phase on remote devices. Since the same devices can be used by multiple application, each with a different context definition, this needs to be studied further. Heterogeneity also implies that not all the devices can provide the same contextual information, thus another path to explore would be to deal with partial contextual information. Finally, we have introduced in this study measurements for the discovery phase, our ongoing work will now focus on contextual request and asynchronous event handling.

References

[1] A. Bandara, T. R. Payne, D. de Roure, and G. Clemo. An ontological framework for semantic description of devices. In *3rd International Semantic Web Conference (ISWC)*, Hiroshima, Japan, November 2004.

[2] FIPA. Fipa device ontology specification, December 2002.

[3] F. Jammes and H. Smit. Service-oriented paradigms in industrial automation. *IEEE Transactions on Industrial Informatics*, 1(1):62–70, February 2005.

[4] J. Kuck and M. Gnasa. Context-sensitive service discovery meets information retrieval. In *Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, pages 601–605. IEEE Computer Society, 2007.

[5] D. Lingrand, S. Lavirotte, and J.-Y. Tigli. Selection using non symmetric context areas. In *Workshop on Context-Aware Mobile Systems (CAMS)*, volume LNCS 3762, pages 225–228, Agia Napa, Cyprus, October 2005. OnTheMove Federated Conferences (OTM'05), Springer.

[6] M. MacKenzie, K. Laskey, F. McCabe, P. Brown, and R. Metz. Reference model for service oriented architecture 1.0. Technical Report wd-soa-rm-cd1, OASIS, February 2006.

[7] J. Pauty, P. Couderc, and M. Banâtre. Synthèse des méthodes de programmation en informatique contextuelle. Technical Report 1595, IRISA, January 2004.

[8] P.-G. Raverdy, O. Riva, A. de La Chapelle, R. Chibout, and V. Issarny. Efficient context-aware service discovery in multi-protocol pervasive environments. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*. IEEE Computer Society, 2006.

[9] L. Seinturier. Composition of contracts for the reliability of service-oriented architectures (faros). Technical report, EDF R&D, FT R&D, I3S, IRISA, LIFL laboratories, May 2007.

[10] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison. Bridging physical and virtual worlds with electronic tags. In *SIGCHI conference on Human factors in computing systems: the CHI is the limit (CHI)*, pages 370–377, Pittsburgh, Pennsylvania, USA, 1999.

[11] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, September 1991.