Multi-level context adaptation in the Web of Things

Mehdi Terdjimi

Multi-level context adaptation in the Web of Things

Mehdi Terdjimi

Université de Lyon, LIRIS
Université Lyon 1 - CNRS UMR5205
F-69622, France
mehdi.terdjimi@liris.cnrs.fr

Abstract. The Web of Things (WoT) aims at connecting things to applications using web technologies, on top of the Internet of Things. WoT applications are distributed and gather different levels of abstraction. They must be scalable and adapt to dynamic changes in their environment. The question we explore in the scope of my PhD thesis is: how can we deal with context in WoT applications? Our objective is to enable scalable multi-level context-aware adaptation for WoT applications. We intend to build models to describe context and reason about it. First, we have studied related work to identify a set of contextual levels and dimensions and have proposed semantic models suitable for several adaptation tasks in WoT applications. Second, we designed and implemented an architecture that distributes some adaptation tasks onto the client side, to improve reasoning scalability.

1 Introduction

The Internet of Things (IoT) aims at connecting devices (i.e. sensors and actuators) to the Internet, to share information using various protocols. The Web Of Things (WoT) builds upon the IoT, where connected devices ("things") rely on Web technologies and standards to break through silos and allow interoperability in pervasive applications [15], e.g. by making use of semantics.

In ubiquitous computing, context highly impacts application behavior and is composed of various pieces of information. Our research question is: how can we deal with context in WoT applications? And subsequently, how can we model context in a generic yet efficient way: on one hand, to harvest contextual data from different sources and build coherent and reliable models? On the other hand, how to actually perform the adaptation process for different purposes, required by WoT applications? To answer these questions, we propose in this thesis the concept of multi-level adaptation.

1.1 Context of the work: the ASAWoO project

In our work as part of the ASAWoO project[1], we propose the concept of avatar to augment and represent physical objects in the virtual world. The avatar of a

[1] http://liris.cnrs.fr/asawoo/

physical objects exposes its functionalities as RESTful resources. In some cases, some features of the object might be disabled for technical reasons such as physical constraints, or due to security, privacy, or other policies. The decision to make features available or not depends on context. To enable adaptation of connected devices to the changes affecting them, appropriate context modeling is required.

1.2 Challenges and motivations

In this work, we aim at using Web standards, such as service sharing protocols (through REST architectural style) and semantic Web technologies (RDF, OWL, SPARQL). The current challenges are:

1. To design generic, semantically-annotated context models for WoT applications, based on the state of the art, to allow context reasoning and adaptation. This will allow interoperability among heterogeneous contextual data sources, such as sensors of the object, external information gathered through Web services and application domain knowledge.
2. To enable multi-level adaptation, as WoT applications cover multiple abstractions, domains and needs. The adaptation would be realized by an appropriate engine, designed in two parts. The first part would be in charge of the extraction of relevant information (depending on the chosen adaptation type). The second part would be the adaptation engine itself.
3. To provide a scalable adaptation engine, in view of the increasing number of heterogeneous connected devices. A modularization of the reasoning steps would allow their distribution between the avatar and its clients. Each step could be executed on the client if its computing resources are sufficient.

In Section 2, we propose a state of the art on context modeling, followed by a state of the art on mobile/client reasoning. We propose in Section 3 a generic, flexible and scalable context model that constitutes our approach. A multi-level semantic adaptation process is presented and evaluated, with a method that locates the reasoning steps between the client and the WoT infrastructure in Section 4. In Section 5, we discuss the results and give the perspectives and future work.

2 Related work

In this section, we overview related work in the field of context modeling to help us build context models. We also study related work in the domain of mobile reasoning to help us build our adaptation solution.

2.1 Context modeling

Former definitions of context are generic and define several dimensions such as Location, Environment, Time, Activity, and User (Schilit and Theimer [26], Pascoe [21], Dey [10], Schmidt [27]). Context models used in IoT applications are

close to the former, but adapted to particular needs [22]. Some of them focus on physical aspects by using context dimensions related to the presence, the activity and the state of entities (i.e. people and devices) in some location, at a certain time [25, 11, 27, 35, 8, 1]. Some works use network context to provide efficient routing and disruption-tolerance [13, 18, 20, 32, 23]. In the field of social computing, context models have been designed for different purposes. To facilitate user interactions with the application [6, 33], or to improve organization within multi-agent systems [4, 3, 2, 5]. There are also works that separate application architecture information and business logic [16, 7, 19, 12], or the device and its physical properties [31, 30]. Another popular usage of context is related to content adaptation, which could be media [34] or, more recently, linked data [9].

The designed context models are specific to the application. But none of them completely rely on the Web, nor perform adaptation on multiple abstraction levels. Thus, as far as we know, there is no multi-level context model for WoT applications.

2.2 Mobile reasoning

When it comes to reason about context to perform adaptation, client-side reasoning may be a solution to address scalability concerns that arise with high numbers of simultaneous requests. But even if client processing resources augment at a fast pace, they remain heterogeneous and in some cases, too limited to execute heavy calculation processes. Thus, adaptivity and flexibility depending on the client's resources are necessary.

The following approaches aim at optimizing the reasoning process for resource-constrained devices. Different ways have been envisioned, from axiom-template rewriting (Kollia and Glimm [17]), to the Triple Pattern Fragments approach (TPF) which relies on intelligent clients that query TPF servers to address the problem of scalability and availability of SPARQL endpoints. However, the use of a server is always necessary. Concerning mobile reasoners, some are based on first-order logic (FOL) (KRHyper [28]), or description logics (DL) (Mine-ME 2.0 [24], $\mathcal{EL}+$ Embedded Reasoner [14]). But KRHyper is not designed for DL, and [24, 14] do not provide a web client access. An approach to embed a reasoner in mobile devices is to rely on web standards and run it in a web browser in Javascript. EYE[2] is a Node.js[3]-compatible reasoner, limited to FOL, which only runs on the server-side. Based on the JSW Toolkit, OWLReasoner[4] allows client-side processing of SPARQL queries on OWL 2 EL ontologies. As far as we know, it is the only full-Javascript OWL 2 EL reasoner that can be used offline in a web client, though its SPARQL engine is limited to basic rule assertions.

[2] http://reasoning.restdesc.org/
[3] https://nodejs.org/
[4] https://code.google.com/p/owlreasoner/

3 Approach

Our approach is based on the research questions raised in Section 1. First we plan to model each abstraction level based on the dimensions identified in the state of the art. As our context model will be described in an ontology, we consider an abstraction as a context state corresponding to a graph part. An example is depicted in Figure 1. In this example, the question *"Which communication protocols can be used?"* will be queried on the graph which contains the state `(Location: home, Security: Level_1, Time: Evening)`. In some cases, a dimension has no use in the abstraction (e.g. the Gender in the Communication level). Secondly, we plan to model the context of the WoT application state, in order to provide adaptation of the reasoning task. To do this, we evaluate our implementation in Section 4 to identify which parameters to model.

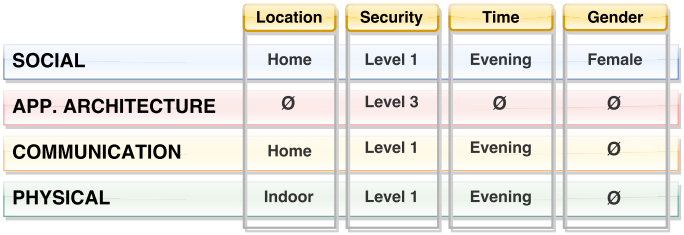

Fig. 1. Example illustrating our approach

The overall method is: 1) to generate graphs from the context models, 2) to query these graphs in SPARQL, in order to retrieve each possibility given a context state, and 3) to add a rule engine that drives the reasoning process. Each step would be separated, to be deferred in the client side. Thus, we propose in the next Section an implementation to reach these goals.

4 Implementation and evaluation

We implemented an architecture that allows the modularization of the reasoning steps and client-side code migration. We proposed in [29] the HyLAR prototype[5] (for Hybrid Location-Agnostic Reasoning), which is a lightweight, modular and adaptive architecture developed in Javascript for hybrid client/server side reasoning, based on OWLReasoner. It allows client-side processing of SPARQL queries on OWL 2 EL ontologies, and consists in the separation of JSW modules that perform ontology classification (JSW Classifier), ontology and SPARQL query parsing (JSW Parser) and reasoning (JSW Reasoner)[6]. JSW modules are

[5] The prototype is available at http://dataconf.liris.cnrs.fr/owlReasoner/

[6] Originally, OWLReasoner rewrites the aBox into a relational database and SPARQL queries into SQL queries. This choice is not ours and will not be discussed here.

packaged as Node.js modules and AngularJS[7] services. This way, they can be executed on either the server or client. On the client side, the reasoner modules can be embedded either in a regular angular service, or in a web worker. As the main service is totally agnostic about the location of the reasoning modules, security and privacy concerns arise, but these aspects are subject to future work.

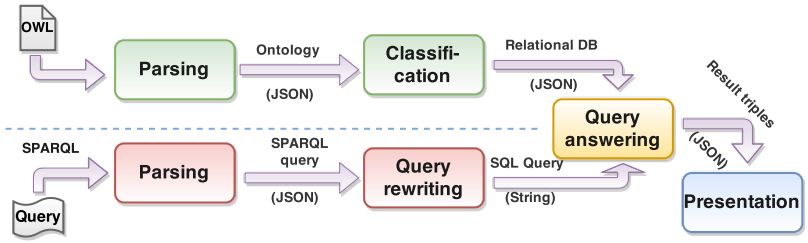

Fig. 2. OWLReasoner's reasoning steps

The goal of the evaluation we propose in [29] is to identify the parameters affecting the reasoning task's processing time, and to find an optimal configuration. We calculated the overall reasoning process request-response times in three situations: full server-side, full client-side and hybrid (server-side parsing and classification, and client-side query processing). For the latter two variants, client-side parts are evaluated both with and without web worker. We assume that scripts and ontologies are available on the server. Each scenario is evaluated with these initial parameters and tools:

- Ontology sizes: ontology A has 1801 class assertions and 924 object property assertions, and B has 12621 class and no object property assertions)[8]. These datasets are conference keywords gathered from DataConf[9].
- Network status: requesting locally or in high latency conditions (around 150ms, using Clumsy 0.2[10]).
- Client capabilities: a Dell Inspiron (i7-2670QM CPU @ 2.20GHz, which also hosts the Node.js server), a Nokia Lumia 1320 (Snapdragon S4 @ 1700 MHz) and a Samsung Galaxy Note (ARM cortex A9 Dual-Core @ 1,4 GHz).

In Tables 1 and 2, [Q] is the time for the client's request to reach the server, [P] is the processing time and [R] is the time for the server response to reach the client. Some parts of these steps/patterns are considered immediate and noted in the result tables as not applicable.

[7] http://www.angularjs.org
[8] Due to OWLReasoner query engine limitations that does not currently allow querying individuals nor data property assertions, our evaluations are limited to class and object property assertions.
[9] http://dataconf.liris.cnrs.fr/
[10] http://jagt.github.io/clumsy/

Ontologies A / B	**[R0]**	**[Q1]**	**[R1]**	**[Q2]**	**[R2]**	**[Q3]**	**[R3]**
Remote server	334	54	110 / 275	119 / 120	167 / 647	146 / 154	61 / 85

Table 1. Network delays (in ms)

Ontologies A / B	**[P2] (no worker)**	**[P2] (worker)**	**[P3] (no worker)**	**[P3] (worker)**
Inspiron (Chrome)	790 / 27612	764 / 26464	28 / 101	24 / 88
Lumia (IE)	1989 / 54702	1883 / 53801	156 / 198	144 / 185
Galaxy Note (Firefox)	2954 / 81255	2872 / 79752	465 / 2988	440 / 2872
Server (Node.js)	780 / 20972	n/a	35 / 37	n/a

Table 2. Classification [P2] and reasoning [P3] times (in ms)

5 Discussion and future directions

5.1 Analysis of evaluation results

As expected, Table 2 shows that the server has the best results for the classification processing time and can use caching. Even if the raw ontology is faster to load than the classification results, loading scripts and data on the client is much faster than performing the same classification step on each client. Thus, it makes no sense to migrate heavy calculations onto clients, rather than pre-calculating them on the server and caching results. More generally, for M clients and N queries/client, we calculate each configuration calculation times as follows[11]:

- Full server-side: $P2_{server} + M \times N \times (Q3 + P3_{server} + R3)$
  We group network (Q, R) and application (P) statuses as the full process is server-side.
- Full client-side: $M \times (R0 + Q1 + R1) + P2_{client} + N \times P3_{client}$
  Q, R and P fully depend on the client. They are therefore difficult to estimate in comparison to the full-server configuration.
- Hybrid: $P2_{server} + M \times (R0 + Q2 + R2) + N \times P3_{client}$
  Client P estimation is easier as it only concerns the query-answering process.

We identify the following parameters affecting the reasoning task: the number of clients (M) and queries per client (N), the network status (Q and R), and the ontology size and computing resources (P).

5.2 Lessons learned and future directions

There is no optimal configuration, and choosing a location for each step is a complex task as it is context-dependent. In our evaluation, we identified different parameters that affect processing times. They are crucial for the adaptation process and to respond to our research questions: how can we model the context 1) to allow the inclusion of the context elements needed by the application and 2) to perform the adaptation process for different purposes, specific to WoT applications? Our future work includes taking users' privacy into account, and

[11] Server-side classification (performed once and then cached) and client-side calculations (performed in parallel) are only counted once.

describing our context model in an ontology. Graphs corresponding to a particular context state would be generated and queried. We plan to reuse an existing adaptation engine that suits our needs for the reasoning process.

References

1. Arias, M.: Context-Based Personalization for Mobile Web Search. Context pp. 33–39 (2008)
2. Bazire, M., Brézillon, P.: Understanding context before using it. Modeling and using context (2005)
3. Brézillon, P., Pomerol, J.: Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain pp. 1–33 (1999)
4. Brézillon, P.: Context in Artificial Intelligence: II. Key elements of contexts. Computers and artificial intelligence pp. 1–27 (1999)
5. Bucur, O., Beaune, P., Boissier, O.: Representing context in an agent architecture for context-based decision making. Proceedings of the Workshop on ... (2005)
6. Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '08 p. 875 (2008)
7. Chaari, T., Laforest, F., Flory, A., Einstein, A.A., Cedex, V.: Adaptation des applications au contexte en utilisant les services web. Proceedings of the 2nd French-speaking conference on Mobility and uibquity computing - UbiMob '05 (2005)
8. Coppola, P., Mea, V.D., Di Gaspero, L., Menegon, D., Mischis, D., Mizzaro, S., Scagnetto, I., Vassena, L.: The context-aware browser. IEEE Intelligent Systems 25(1), 38–47 (2010)
9. Costabello, L., Villata, S., Gandon, F.: Context-aware access control for rdf graph stores. In: ECAI. pp. 282–287 (2012)
10. Dey, A.K.: Understanding and using context. Personal and ubiquitous computing 5(1), 4–7 (2001)
11. Dey, A.K., Salber, D., Abowd, G.D., Futakawa, M.: The conference assistant: Combining context-awareness with wearable computing. In: Wearable Computers, 1999. Digest of Papers. The Third International Symposium on. pp. 21–28. IEEE (1999)
12. Gensel, J., Villanova-Oliver, M., Kirsch-Pinheiro, M.: Modèles de contexte pour l'adaptation à l'utilisateur dans des systèmes d'information web collaboratifs. In: Workshop from" 8èmes journées francophones". Sophia-Antipolis, France (2008)
13. Gold, R., Mascolo, C.: Use of context-awareness in mobile peer-to-peer networks. In: Distributed Computing Systems, 2001. FTDCS 2001. Proceedings. The Eighth IEEE Workshop on Future Trends of. pp. 142–147. IEEE (2001)
14. Grimm, S., Watzke, M., Hubauer, T., Cescolini, F.: Embedded $\mathcal{EL}+$ reasoning on programmable logic controllers. In: The Semantic Web–ISWC 2012, pp. 66–81. Springer (2012)
15. Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the internet of things to the web of things: Resource-oriented architecture and best practices. In: Architecting the Internet of Things, pp. 97–129. Springer (2011)
16. Kirsch-Pinheiro, M., Gensel, J., Martin, H.: Representing context for an adaptative awareness mechanism. In: Groupware: Design, Implementation, and Use, pp. 339–348. Springer (2004)

17. Kollia, I., Glimm, B.: Optimizing sparql query answering over owl ontologies. arXiv preprint arXiv:1402.0576 (2014)
18. Mascolo, C., Capra, L., Emmerich, W.: Mobile computing middleware. In: Advanced lectures on networking, pp. 20–58. Springer (2002)
19. Munnelly, J., Fritsch, S., Clarke, S.: An aspect-oriented approach to the modularisation of context. In: Pervasive Computing and Communications, 2007. PerCom'07. Fifth Annual IEEE International Conference on. pp. 114–124. IEEE (2007)
20. Musolesi, M., Mascolo, C.: Car: context-aware adaptive routing for delay-tolerant mobile networks. Mobile Computing, IEEE Transactions on 8(2), 246–260 (2009)
21. Pascoe, J.: Adding generic contextual capabilities to wearable computers. In: Wearable Computers, 1998. Digest of Papers. Second International Symposium on. pp. 92–99. IEEE (1998)
22. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: A survey. Communications Surveys & Tutorials, IEEE 16(1), 414–454 (2014)
23. Raverdy, P.G., Riva, O., de La Chapelle, A., Chibout, R., Issarny, V.: Efficient context-aware service discovery in multi-protocol pervasive environments. In: Mobile Data Management, 2006. MDM 2006. 7th International Conference on. pp. 3–3. IEEE (2006)
24. Ruta, M., Scioscia, F., Loseto, G., Gramegna, F., Ieva, S., Di Sciascio, E.: Minime 2.0: powering the semantic web of things. In: 3rd OWL Reasoner Evaluation Workshop (ORE 2014)(jul 2014) (2014)
25. Schilit, B.N., Adams, N., Gold, R., Tso, M.M., Want, R.: The parctab mobile computing system. In: Workstation Operating Systems, 1993. Proceedings., Fourth Workshop on. pp. 34–39. IEEE (1993)
26. Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. Network, IEEE 8(5), 22–32 (1994)
27. Schmidt, A.: Ubiquitous computing-computing in context. Ph.D. thesis, Lancaster University (2003)
28. Sinner, A., Kleemann, T.: Krhyper–in your pocket. In: Automated Deduction–CADE-20, pp. 452–457. Springer (2005)
29. Terdjimi, M., Médini, L., Mrissa, M.: HyLAR: Hybrid Location-Agnostic Reasoning. In: ESWC Developers Workshop 2015. pp. 1–6. Portoroz, Slovenia (May 2015)
30. Truong, H.L., Dustdar, S., Baggio, D., Corlosquet, S., Dorn, C., Giuliani, G., Gombotz, R., Hong, Y., Kendal, P., Melchiorre, C., et al.: Incontext: A pervasive and collaborative working environment for emerging team forms. In: Applications and the Internet, 2008. SAINT 2008. International Symposium on. IEEE (2008)
31. Truong, H.L., Juszczyk, L., Manzoor, A., Dustdar, S.: ESCAPE–an adaptive framework for managing and providing context information in emergency situations. Springer (2007)
32. Wei, Q., Farkas, K., Prehofer, C., Mendes, P., Plattner, B.: Context-aware handover using active network technology. Computer Networks 50(15), 2855–2872 (2006)
33. Xiang, B., Jiang, D., Pei, J., Sun, X., Chen, E., Li, H.: Context-aware ranking in web search. Sigir 2010 p. 451 (2010)
34. Yu, Z., Zhou, X., Zhang, D., Chin, C.Y., Wang, X., et al.: Supporting context-aware media recommendations for smart phones. Pervasive Computing, IEEE 5(3), 68–75 (2006)
35. Zimmermann, A., Lorenz, A., Oppermann, R.: An operational definition of context. In: Modeling and using context, pp. 558–571. Springer (2007)