Dynamic Deployment and Reconfiguration of Intelligent Mobile Cloud Applications Using Context-driven Probabilistic Models

Nayyab Zia Naqvi, Davy Preuveneers and Yolande Berbers
iMinds-DistriNet, KU Leuven, Belgium
Email:{nayyab.naqvi, davy.preuveneers, yolande.berbers}@cs.kuleuven.be

***Abstract*—Today's mobile devices with advanced computing and storage resources are encouraging a whole new class of applications geared towards context-aware intelligence. However, for continuous processing of context-processing algorithms in interactive human-computing interfaces and augmented-reality experiences, these devices lack resource demands. Most of these intelligent applications rely on cloud paradigm for on demand computing, memory and storage resources. While combining both computing paradigms, finding the best strategy to deploy and configure intelligent applications, is not straightforward. In this paper, we analyse the challenges and requirements for the dynamic deployment of intelligent applications in such a federated setting. Additionally, a framework that leverages Dynamic Decision Network (DDN) is presented for decision making. DDN-based models deal with the presence of uncertainty and the partial observability of the context information, as well as the temporal effects of decisions to ascertain the Quality-of-Service and Quality-of-Context requirements. Our initial experiments with the framework demonstrate the feasibility of our approach and potential benefits to automatically make the best decision in the presence of a changing environment addressing the runtime variability.**

Keywords–dynamic deployment; probabilistic models; mobile cloud computing;intelligent applications.

I. INTRODUCTION

With mobility and context-awareness [1], an ecosystem that can better connect the virtual world with the physical world is propagating. This propagation has multiplied the use of mobile devices in smart homes and offices, smart health, assisted living, smart cities and transportation. Research and development of context-aware intelligence aim at the ways to teach our devices about our environment narrowing the gap between us, our devices and the environment. Moreover, a relentless spurt of research activities in Mobile Cloud Computing (MCC) [2] aim to overcome the limitations of state-of-the-art mobile devices for intelligent applications. These cutting-edge applications require continuous processing and high-rate sensors' data to capture the users' context, such as their whereabouts and ongoing activities as well as the runtime execution context on the mobile devices. These devices use cloud resources to cope with the ever-growing computing and storage demands for its applications as a key enabler to run more demanding or long running tasks and applications.

The most viable technique in MCC is to dynamically adapt the mobile application by outsourcing the resource-intensive tasks to the cloud servers. Cloud computing can potentially save resources for mobile users [3][4]. Contrary to mobile devices, cloud computing provides plentiful storage and processing capabilities. This federated design could be applied to almost any application and has been shown to improve both the speed and energy consumption for non-trivial computations [5].

Nonetheless, not all applications are energy efficient when migrated to the cloud. Additionally, a lot of contextual information is sensed and captured on mobile devices. To deploy and initialize the desirable components automatically in this federated environment, involves several trade-offs with respect to the Quality-of-Service (QoS) [6] and the Quality-of-Context (QoC) [7]. A modular design philosophy for intelligent applications enables a more optimal deployment and performance when leveraging cloud technology. However, attaining the best deployment and configuration strategy for intelligent applications is not straightforward. In the core of such a distributed environment, adaptation decision of what to run where and when is non-trivial. The requirements of semantic knowledge and intelligence, the resource characteristics of the application components and low-level monitoring of the platforms used for deployment in terms of processing power, bandwidth, battery life and connectivity makes this decision highly dynamic. The main causes of this dynamism are runtime uncertainty and erratic nature of the context information [8], significantly impacting deployment decisions and performance. This context uncertainty can lead to annoying or incorrect decisions and can negatively impact the ability to reliably predict future contexts for proactive decision making as well. Furthermore, the system should be made aware of the impact of its decisions over time to optimize the runtime deployment and learn from its mistakes as humans do. The decision making needs to be flexible enough to ascertain the quality of its own decisions.

We present a framework designed to support the modular development and deployment of intelligent applications within the setting of MCC. We have adopted a probabilistic model based on DDNs for optimal decision making under evolving context of the runtime environment of a mobile device. DDN is an emerging research topic, and researchers are investigating its use in the area of self-adaptation for autonomous systems in several domains. Our major contribution in this paper is an approach of context-driven decision making using DDNs for dynamic deployment of intelligent applications, dealing with the dynamism and the partial observability of the context, as well as the temporal effects of the decision. Our model is able to learn deployment trade-offs of intelligent applications and capable of learning from earlier deployment or configuration mistakes to better adapt to the setting at hand. We have worked out our approach to an augmented-reality based use case where a DDN model decides and learns for the deployment of application components. Our experiments demonstrate the feasibility of the approach and potential benefits to automatically

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-352-0

make the best decision in the presence of changing situations or circumstances.

This paper is structured in six sections. In Section II, we give an overview of related work in MCC and discuss the gaps in state-of-the-art for federated deployments. Section III highlights the requirements and objectives of our use case scenario. Section IV provides a brief account of our proposed federated framework and the details about our approach of learning the trade-offs for dynamic deployments using a probabilistic decision model, mitigating the influence of runtime uncertainty. Finally, after evaluating our approach applied on our use case scenario in Section V, the paper concludes and offers a discussion of topics of interest for future work.

II. Background and related work

Although outsourcing the computation to the cloud can be beneficial in terms of QoS [6]. This distribution is never clear-cut in the scenario of intelligent systems considering trade-offs with respect to QoS and QoC [9]. The question is, where do you draw the line? What should be run on the cloud and what work should be done by the mobile device? The easiest option is to have everything stay local and not use the cloud or Internet at all, but as mentioned earlier, this could lead to bad performance of the application. Since the cloud consists of so many computers and has so much processing power, it is tempting to just decide to do almost all the work online. However, we can not forget that communication with the cloud has a price as well, economically and in terms of time and energy. In this case, multiple conflicting objectives affect the decisions.

Fortunately, researchers have developed a selection of systems that allow the application to make the partitioning decision while it is running, to provide the user with the best experience possible. CloneCloud [5] clones the entire set of data and applications from the smart phones to the cloud and selectively execute operations on the clone. However, for the continuous nature of context data, the CloneCloud [5] approach fails in a way that the dataset is not predetermined and needs to be processed in real-time for context-aware intelligence. Even more impressive, some of these platforms can let applications enjoy the best of computation offloading by only making minor changes, even when they have already been built without cloud computing capabilities [10]. Cloudlets [11] is another approach to achieve QoS but it does not provide an answer for the distribution of processing, storage, and networking capacity for each cloudlet. How to manage policies for cloudlet providers to maximize user experience while minimizing cost, security and trust are also open issues in order to adopt it for practical systems. Weblet [12] dynamically decides whether a weblet; a specialized form of HTTP-based web service interface, runs on the mobile device or the cloud with the help of Naive Bayesian learning techniques to find the optimal weblet configuration at runtime. Research is being done to investigate the usefulness of clouds accessing other clouds to reduce time and energy spent messaging back and forth between the mobile device and different cloud services [13].

Restating the obvious, intelligent applications have to take into account the runtime uncertainty in context data as context sources are dynamic in nature. They can disappear and re-appear at any time and context models change to include new context entities and types. The properties of context sources and context types can change randomly and the uncertainty can vary too. Fenton and Neil [14] have used Bayesian networks for predictions of the satisfaction of non-functional aspects of a system. Esfahani et al. [15] employ fuzzy mathematical models to tackle the inherent uncertainty in their GuideArch framework while making decisions on software architectures. Dynamic configuration of service oriented systems was investigated by Filieri et al. [16]. In contrast to our model, they used Markov models to investigate the decision making under uncertainty and quality requirements. Many works use utility functions to qualify and quantify the desirability of different adaptation alternatives. Bencomo and Belggoun [17] used DDNs to deal with the runtime uncertainty in self-adaptive systems. But these works are QoS-based, applied in different domains for resource allocation [18] and typically in component-based mobile and pervasive systems such as Odyssey [11] and QuA [19]. Markovian decision models still lack the tractability [20] for complex decisions under uncertainty.Moreover, the current state of the system has to be known, in order to use Markovian decision models. Several intelligent applications are pretty lightweight, but others require a lot of computational effort (e.g., for prediction) or require analysis of large amounts of data (e.g., for pattern analysis).

Our approach addresses the concern by identifying the deployment and performance trade-offs for outsourcing data and computation by continuously learning and adapting under multiple conflicting QoS and QoC objectives. If the decision is affirmative, required functionality is executed as a composition of loosely-coupled services on the cloud. Otherwise, lightweight component-based equivalents of these services are executed on the mobile.

III. Use case scenario

The use case considered in this paper is an AR mobile application called *Smart Lens*. Mobile Phones have been the most common platform for Augmented Reality (AR), ever since they were equipped with cameras. A mixture of AR and context-aware intelligence is often found in applications that aim to help the user explore certain places, be it cities, expos, museums or malls. In many cases, it makes the phone act as a camera but adds extra information next or onto objects that appear on screen. In short, mobile AR allows the devices to recognise objects as the user would, by seeing them.

A. Working procedure and requirements

By pointing to an electrical appliance, *Smart Lens* enables the users to be effortlessly aware of the power consumption of that appliance on top of that image. In addition to the near-real-time power consumption of the appliances, the *Smart Lens* application allows the user to get a detailed overview of the impact of any particular device compared to others in their electricity bill. Also, by storing the historic data it allows the user to monitor the performance of a device over time and determine whether it is time to replace a device with a more energy-efficient alternative. The modular design have been adopted for data and control processing on mobile and cloud ends to achieve a flexible distributed deployment. It simplifies redeployments and reconfigurations significantly.

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-352-0

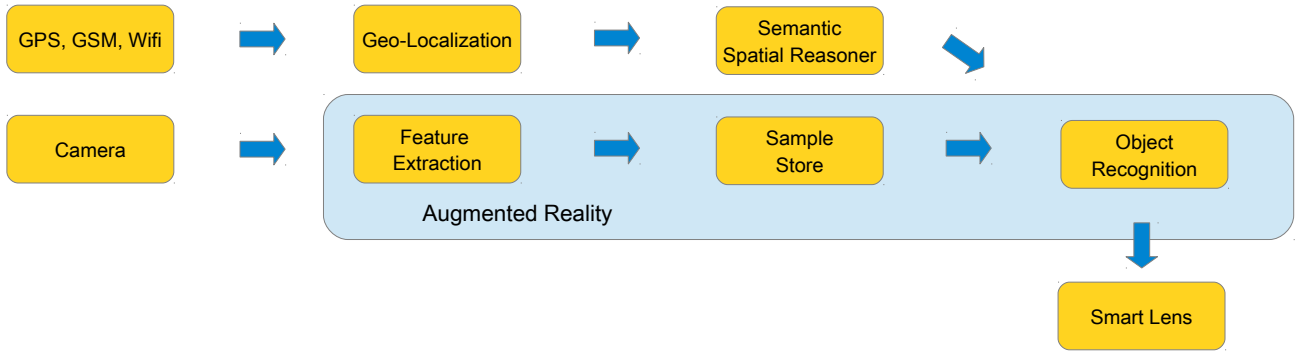

Figure 1. Deployment view with modular design philosophy for the components of *Smart Lens* application.

Figure 1, provides an overview of the composition of the components for our use case.

The major functional and non-functional system requirements for the use case are:

1) The system should be able to identify the electronic appliance, preferably with a markerless [21] approach.
2) The system should be able to capture the real-time context of the user and his environment.
3) The system must be adaptive at runtime to optimize the resource consumption of the application by outsourcing few components to the cloud.
4) The system should detect the runtime context of the mobile device i.e., CPU usage, memory consumption and battery usage. It should take into account the runtime evidence to mitigate contextual uncertainty and take the most rational decision.
5) The system should select the deployment strategy with respect to QoS & QoC requirements from the application's perspective, such as L_{min} for *Minimum Latency* and R_{max} for *Maximum Reliability* in terms of performance.

B. Objectives

Many opportunities for optimization exist as there are several distributed deployments of the application components and different configurations per component possible. The challenge is to find and analyse different optimization trade-offs in a federated environment of MCC, each characterized by varying sensing, communication, computation and storage capabilities. Moreover, mitigate the influence of runtime uncertainty in context and its quality.

A smart phone camera consumes its energy at a higher rate. The application might be able to identify an appliance but there are a lot of other appliances in a household, and user can point on them from any direction, so a conclusion based only on a single view of an appliance from one particular angle, is many times more likely to be incorrect. The *Semantic Spatial Reasoner* helps to maintain the reliability of the application. On top of that, *Feature Extraction* and *Object Recognition* techniques require a relatively large amount of processing time, which would not only drain battery further but causes the application and the phone as a whole to slow down as well.

The easiest option is to deploy every component locally on the mobile device and not use the cloud at all, but this could lead to bad performance of the application. Since the cloud consists of so many computers and has so much processing power, it is tempting to just decide to do almost all the work online. However, we can not forget that communication with the cloud has a price as well, economically and in terms of time and energy.

Based on the above mentioned requirements and shortcomings, we contrive the methodology for the well-informed decision making for dynamic deployment as follows:

1) A traditional QoS & QoC requirements gathering to identify and model the required quality attributes at design time. It is domain specific and involves the type of context being utilized.
2) Runtime support to detect a change in QoC in a particular context type and to measure its impact on other context types before making any decision.
3) Runtime support to detect a change in QoS before making any decision.
4) Enforcement of QoS & QoC policies or ways to ensure these requirements while making a decision.

In the next section, we will explain our context-driven dynamic deployment framework and its learning mechanism using probabilistic models to achieve well-informed decision making with the above described features.

IV. Context-driven dynamic deployment approach

Our framework consists of a loosely-coupled context-processing system, where each component has a dual implementation: an implementation for the mobile and a full-fledged scalable service equivalent running in the cloud. As shown in Figure 2, mobile hosts a *Dynamic Adaptation Module*, i.e., a client-side component of decision module to adapt the deployment configuration of the application. However, the cloud environment hosts a *Service Adaptation Module* which aims to optimize the runtime deployment of the required components and acts as an entry point for the adaptation module on mobile client. This module receives raw or pre-processed context data (including the type of the content and the identity of the source) and forwards it to a publish/subscribe subsystem so that interested parties (i.e., the subscribers) receive context updates.

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-352-0

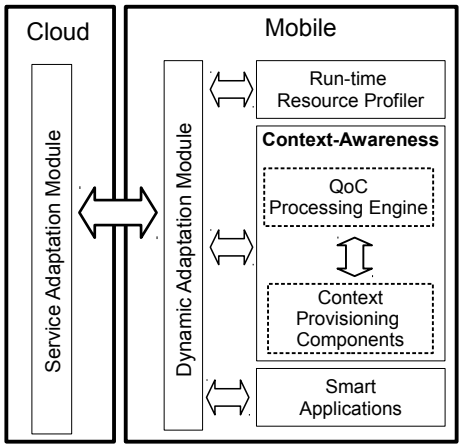

Figure 2. A blueprint of federated MCC framework and its dynamic deployment modules.

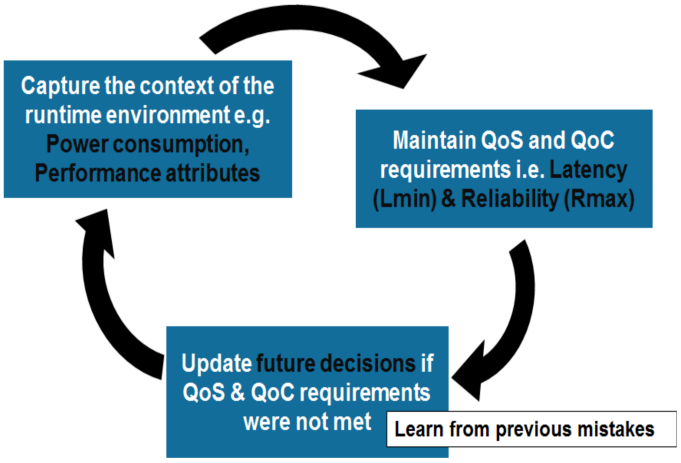

Figure 4. Enactment of dynamic decision making where system learns with it mistakes.

A. Deployment and reconfiguration decision making

Deployment Adaptation Module takes the decision of how to split responsibilities between mobile devices, applications and the framework itself. It should be able to decide for which use cases the cloud is better and for which ones a cloud based deployment does not bring any added value. Our decision making approach for redeployment and reconfiguration is explained in steps mentioned below:

Information discovery and selection – The framework discovers and explores application's runtime environment in order to get the context information to work with. Figure 3 shows a taxonomy of the runtime environment of a mobile device. Its resources can affect the QoS requirements and eventually, the decision of dynamic deployment.

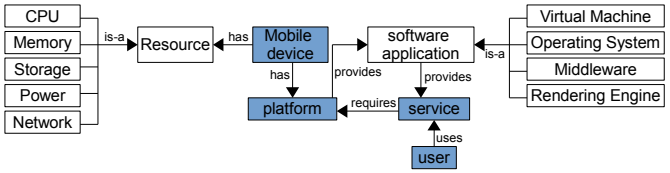

Figure 3. Taxonomy of the runtime environment of a mobile.

Our framework discovers the sensors and context types required for the smart application. Built-in sensors in mobile devices are important to fetch the context data, but the size of the acquired context data varies, depending on the application's objectives. The framework filters acquired context according to specific needs of the application. This selection process can be fairly complex as it may require complex filtering techniques to decide which sensor or device is offering relevant information. QoS and QoC requirements are gathered at this step to bootstrap the decision making. *Runtime Resource Profiler* component deployed on the mobile side, gathers these requirements.

Analysis and decision making – The framework uses discovered and selected information to make the deployment decision or change the configuration or behaviour of the application. For instance, user points his mobile device on an appliance. The image/video frame from the mobile device is captured. The feature points for the image in the frame are extracted. With these feature points , planar objects are then identified by matching the extracted feature points to those of known planar objects in a database. There is little memory because of his running video player. In this situation, a thin client configuration is chosen for *Smart Lens*, delegating *Spatial Reasoning* and *Object Recognition* components to cloud infrastructure. Furthermore, when the user shuts down video player, a context change is raised: free memory on the hand-held becomes high.

Enactment of the decisions – With a thin client configuration in previous step, the framework has to observe the real-time impact of the configuration to maintain QoS requirement of *Minimum Latency*. In order to increase application response time *Smart Lens* is reconfigured with a thick client and caching of data to save power and to become less vulnerable to network instability. As depicted in Figure 4, decision making is a continuous process, where the framework optimizes the application behaviour to reach certain objectives on the basis of the required attributes. When the availability of required resources varies significantly, framework has to decide whether to trigger an adaptation in the form of reconfiguration of the components. It learns from its previous decisions and the available context in order to ascertain the quality of the decision and learn from its own mistakes to achieve better results in future.

B. Learning the deployment trade-offs using DDNs

Conditional probability distributions derived by analysing historical attribute values helped solve stochastic problems in past. Runtime setting for every component is hard to determine in advance due to the dynamic interaction of these components with the environment and the user. Our *Deployment Adaptation Module* takes an advantage of probability theory and statistics to describe uncertain attributes. Probabilistic reasoning allows the system to reach rational decisions even when complete information is not available. Knowledge about runtime uncertainty can be captured by a data structure for probabilistic inference called a Bayesian network (BN). A BN is a Directed Acyclic Graph (DAG) represented by a triplet (N, E, P), where N is the set of chance nodes, E is the set of arcs to represent causal influence of the chance nodes and P is the conditional probability distribution for each chance node.

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-352-0

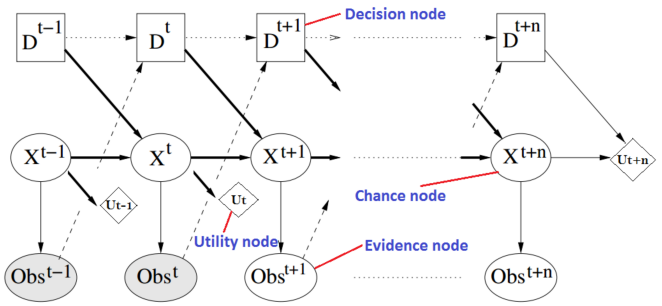

Figure 5. Structure of a DDN with dynamic chance nodes affecting utility nodes with decision nodes and evidences [22].

A Decision Network is a BN that also includes a set of decision and utility nodes. Utility nodes express the preferences among possible states of the world in terms of a subset of chance nodes and decision nodes. A probability-weighted expected utility is calculated for each decision given the evidence. To represent variables that change over time, it is possible to use a time-sliced network such that each time-slice corresponds to a time point. A DDN [17] is used for the enactment of the decisions that change over time influenced by dynamic states and preferences. To model the effectiveness of reconfiguration over time, decisions are modeled using a DDN where each time slice contains an action taken by the system. Figure 5 shows the structure of a general DDN.

DDNs are fed with runtime context of the user and the execution environment, such as CPU or memory usage, remaining battery and network connectivity. Chance nodes can represent runtime context, QoS or QoC requirements for each component of the intelligent application. Evidence nodes are different runtime situations based on the context of the user and parameters of the system. Decision nodes are the actual reconfiguration decision for each component. Utility nodes are influenced by chance nodes and decision nodes to infer the utility of dynamic deployment decision for a component. Utility functions can be used to assign priorities to different QoS and QoC requirements. The utility of redeployment decision is inferred using the QoS and QoC requirements of the application. System will choose the decision with highest expected utility, known as the Maximum Expected Utility (MEU) principle [22].

Our model expresses QoS_i, QoC_i and the context of the mobile device by chance nodes. These chance nodes make a BN with conditional probabilities corresponding to the effects of different actions D_j over QoS_i or QoC_i. Evidence nodes, defined as "Obs" in Figure 5, express the uncertainty factors connected to the chance nodes to take a favorable decision. For each dynamic chance node, utility node expresses the utility function that takes conditional probabilities of QoS or QoC requirements and their priorities into account. The expected utility for each decision is computed with respect to the $P(QoS_i|QoC_i,D_j)$, $P(QoC_i|D_j)$ and a weight for the decision as expressed in Equation (1).

$$EU(D_j \mid QoC_i) = {}_i\Sigma P(QoS_i \mid QoC_i, D_j)U(QoS_i \mid D_j) \quad (1)$$

V. Experimental Evaluation

In order to implement our framework, discussed in Section IV-A, we are using an HTC One X with a 1.5 GHz Quad Core ARM Cortex processor (at about 2.5 MIPS per MHz per core) to run the Android-based *Smart Lens*, augmented reality application, which embeds the Vuforia library to recognize everyday devices of the users in the environment. Our infrastructure runs VMware's open source Platform-as-a-Service (PaaS) offering known as Cloud Foundry on a server with 8 GB of memory and an Intel i5-2400 3.1 GHz running a 64-bit edition of Ubuntu Linux 12.04. A Java-based implementation has been used for *Runtime Resource Profiler* that captures the runtime context of mobile device. We have modeled a DDN-based probabilistic model for the components of our *Smart Lens* use case using JSmile Android API in Genie and Smile environment from Decision Systems Laboratory [23]. Our DDN model is evaluated using Equation (1) for every decision D_j computing the probability-weighted average utility for that decision.

The initial experiments are conducted for the redeployment of *Object Identification* component. Our model has been designed and expanded for 5 time slices as shown in Figure 6. CPU and memory usage are modeled as static chance nodes representing execution context. QoS requirement *Minimum Latency* is modeled as a dynamic chance node influencing the redeployment decision and QoC requirement *Maximum Reliability* is observed in terms of performance. The QoS and QoC requirements both affect utility node. Decision node is the actual reconfiguration decision for each component.

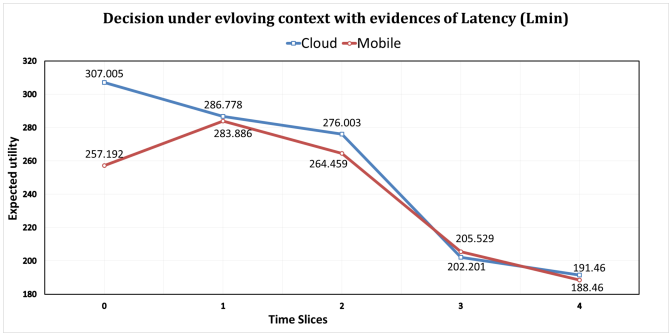

Figure 6. Expected utility for redeployment decision of *Object Identification* component with changing runtime situations.

QoS requirement of latency is kept minimum with respect to a threshold value of 3 seconds. Initial conditional probabilities are defined with respect to QoC requirements where the CPU usage is kept less than 40% and memory usage less than 45 MB. Figure 6 shows that our model learns and adapts according to a change in the context of the runtime environment at each time slice, where each time slice is realized by a ticker initiated from *Runtime Resource Profiler*, whenever the runtime environment changes. At time slice 4, latency is observed and model adapts itself to deploy this component on the mobile device rather than the cloud, according to the required QoS.

VI. Conclusion and Future Work

The optimal strategy to deploy and configure intelligent applications with dynamic and heterogeneous resource availability is not forthright. This deployment of software components has to take into account the resource characteristics of application components and the platforms used for deployment in terms of processing power, bandwidth, battery life and

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-352-0

connectivity. Our modular design philosophy for developing intelligent applications helps to dynamically configure, compose and deploy these components. The overall aim of our work is to intelligently automate the distributed deployment and configuration of the components across the mobile and cloud infrastructures.

In this paper, we have presented a novel approach for dynamic deployment decision making in federated environment of MCC by leveraging DDN to automate decisions in a continuously evolving runtime environment context. DDNs build upon Dynamic Bayesian Networks. However, the latter is only able to learn conditional probabilities based on a dataset, whereas DDNs can quantify the impact of the evidence and the effect of the decisions. Furthermore, by exploiting the utility of deployment decisions, our framework can learn how to automatically improve its decisions in the next iteration or time slice. Our first contribution was a feasibility analysis of incorporating DDNs for decision making, and our experiments have clearly demonstrated the ability of adapting its decision in the presence of evolving situations and an uncertain context of the environment. By incorporating QoS & QoC in our DDNs, we are able to assess the quality of our context-driven decisions, ascertain their quality and update future decisions and corresponding actions according to the outcome and impact. Our preliminary experiments have shown that an intelligent application can achieve optimal deployment for its components, whenever the context is updated. However, the sensitivity analysis in federated environment has to be conducted.

Further work is required towards more systematic techniques for the runtime synchronization of multiple DDN models and to empirically study the scalability of these models. The value of the probabilities that change over time and their impact on alternative decisions can also be of interest. Finally, developing tools to specify the QoC requirements and design a DDN would be certainly very helpful as current tool's support for modelling and using DDNs is fairly limited.

ACKNOWLEDGMENT

This research is partially funded by the Inter University Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund KU Leuven.

REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and ubiquitous computing*, pp. 304–307, Springer, Springer, 1999. [retrieved: March, 2014].

[2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, pp. 1587–1611, 2013. [retrieved: January, 2014].

[3] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?," *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[4] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pp. 4–4, 2010.

[5] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution.," in *HotOS*, vol. 9, pp. 8–11, 2009. [retrieved: Feb, 2014].

[6] D. Chalmers and M. Sloman, "A survey of quality of service in mobile computing environments," *Communications Surveys & Tutorials, IEEE*, vol. 2, no. 2, pp. 2–10, 1999. [retrieved: October, 2013].

[7] T. Buchholz, A. Küpper, and M. Schiffers, "Quality of context: What it is and why we need it," in *Proceedings of the workshop of the HP OpenView University Association*, vol. 2003, Geneva, Switzerland, 2003. [retrieved: June, 2013].

[8] N. Chen and A. Chen, "Integrating context-aware computing in decision support system," in *Proceedings of the International MultiConference of Engineers and computer Scientists*, vol. 1, pp. 359–364, 2010. [retrieved: June, 2013].

[9] N. Z. Naqvi, D. Preuveneers, Y. Berbers, *et al.*, "Walking in the clouds: deployment and performance trade-offs of smart mobile applications for intelligent environments," in *Intelligent Environments (IE), 2013 9th International Conference on*, pp. 212–219, IEEE, 2013. [retrieved: January, 2014].

[10] Y. Zhang, G. Huang, X. Liu, W. Zhang, H. Mei, and S. Yang, "Refactoring android java code for on-demand computation offloading," in *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, pp. 233–248, ACM, 2012.

[11] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009. [retrieved: December, 2013].

[12] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011. [retrieved: December, 2013].

[13] S. Kosta, V. C. Perta, J. Stefa, P. Hui, and A. Mei, "Clone2clone (c2c): Peer-to-peer networking of smartphones on the cloud," in *5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud13)*, 2013. [retrieved: February, 2014].

[14] N. Fenton and M. Neil, "Making decisions: using bayesian nets and mcda," *Knowledge-Based Systems*, vol. 14, no. 7, pp. 307–325, 2001. [retrieved: December, 2013].

[15] N. Esfahani, K. Razavi, and S. Malek, "Dealing with uncertainty in early software architecture," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ACM, 2012. [retrieved: January, 2014].

[16] A. Filieri, C. Ghezzi, and G. Tamburrelli, "A formal approach to adaptive software: continuous assurance of non-functional requirements," *Formal Aspects of Computing*, vol. 24, no. 2, pp. 163–186, 2012. [retrieved: December, 2013].

[17] N. Bencomo and A. Belaggoun, "Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks," in *Requirements Engineering: Foundation for Software Quality*, pp. 221–236, Springer, 2013. [retrieved: June, 2013].

[18] T. Kelly, "Utility-directed allocation," in *First Workshop on Algorithms and Architectures for Self-Managing Systems*, vol. 20, 2003. [retrieved: January, 2014].

[19] S. Amundsen, K. Lund, F. Eliassen, and R. Staehli, "Qua: platform-managed qos for component architectures," in *Proceedings from Norwegian Informatics Conference (NIK)*, pp. 55–66, 2004. [retrieved: March, 2014].

[20] P. C. Da Costa and D. M. Buede, "Dynamic decision making: a comparison of approaches," *Journal of Multi-Criteria Decision Analysis*, vol. 9, no. 6, pp. 243–262, 2000.

[21] E. Ziegler, "Real-time markerless tracking of objects on mobile devices," 2012. [retrieved: December, 2014].

[22] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Englewood Cliffs, 1995. ISBN: 0131038052.

[23] M. J. Druzdzel, "Smile: Structural modeling, inference, and learning engine and genie: a development environment for graphical decision-theoretic models," in *AAAI/IAAI*, pp. 902–903, 1999. [retrieved: March, 2014].

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-352-0