Opportunistic Use of Cloud Computing for Smart Mobile Applications: Realizing Qo*-awareness with Dynamic Decision Networks in Execution

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

***Abstract*—Smart and context-aware Mobile Cloud Computing applications challenge the way Quality-of-Service and Quality-of-Context are maintained when operating with a federated mobile cloud environment. Many resource and performance trade-offs exist for the federated deployment of smart mobile applications. Consequently, finding the best strategy to deploy and configure intelligent applications in this federated environment is a non-trivial task. We analyse the challenges and requirements for the dynamic deployment of smart applications in such a setting and present a quality-aware federated framework for the development and deployment of smart mobile applications to use the cloud opportunistically. Our framework utilizes Qo*-awareness and dynamic adaptability to account for the uncertain conditions given the partially observable context. Experiments with our framework demonstrate the feasibility and the potential benefits to automate the deployment and configuration decisions in the presence of a changing environment and runtime variability.**

Keywords–dynamic deployment; Bayesian models; mobile cloud computing; smart applications; decision support.

I. Introduction

Mobility and context-awareness have multiplied the use of mobile devices in homes or offices, health and cities giving them a *Smart* label. Intelligent and smart applications are gaining popularity day-by-day due to the ease and the control they offer to the user. Continuous advancements in mobile technology are changing our habits and the ways we interact with these mobile applications [1]. This rapidly evolving mobile technology is giving a momentum to the smartness of these devices and a better control of our mobile applications, considering the perspective of the user and his situation [2]. A wide range of sensing, communication, storage and computational resources makes these mobile devices a perfect platform for ubiquitous computing offering pervasive connectivity and a source for context-awareness.

Context-awareness is a congenital characteristic [3] of intelligent applications to the notion of adaptability and smartness. Context [4] is defined as any information that can be used to characterize the situation of an entity, where an entity can be an object, a place or a person relevant to the current scope of the system. Research and development of context-awareness is narrowing the gap between us, our devices and the environment. Context aims to become the fabric of ubiquitous computing. The consequent smart applications behave as the constituents of ubiquitous environments as envisioned by Mark Weiser [5].

The user takes a passive role in context-aware applications and these mobile applications decide on his behalf. These applications require continuous processing and high-rate sensors' data to capture the user's context, such as his whereabouts and ongoing activities as well as the runtime execution context on the mobile device. However, being smart requires being right in an adaptation decision. Context is dynamic in nature and is prone to ambiguity. This uncertainty not only leads to incorrect decisions but also makes proactive decision making impossible, impacting the *Quality-of-Service (QoS)* of the application in a negative fashion. We need certain attributes that signify the adequacy or degree of suitability of the context data. This degree of suitability, often regarded as the *Quality-of-Context (QoC)* [6], can highly affect the adaptation decision.

Despite the continuous improvements in mobile technology, the exponential growth in mobile usage is formidable to cope with the resource limitations. Mobile Cloud Computing (MCC) offers Mobile Cloud Augmentation (MCA) with the aim to empower mobile devices to run more demanding or long running tasks by providing plentiful storage and processing capabilities on cloud servers. Cloud computing [7] offers access to an always connected, decentralized, and abstracted infrastructure of a heterogeneous pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can scale up or down instantly in a controlled fashion. This federated design can be applied to almost any application and has been shown to improve both the speed and energy consumption [8]. Most of the cloud applications follow a thin client philosophy where resource intensive tasks are outsourced to the cloud infrastructure in a brute force fashion.

A modular design philosophy for smart applications is ideal enabling an optimal deployment in MCA. However, attaining the most suitable deployment and configuration strategy for these applications is not always clear and straightforward. The MCA federation for such type of applications is a non-trivial task. On the one hand, we have mobile devices with built-in sensors to sense the context of user. On the other hand, cloud resources can be utilized opportunistically to save resources for mobile users [9][10]. At the core of such a distributed environment, a decision of *what* to run *where* and *when* is gruelling under a changing runtime execution context. Figure 1 shows an overview of the federation in MCA for context-aware applications. There can be multiple deployment strategies for a smart application both at design time and at runtime. However, it is not clear which component should be deployed where, as the context changes at runtime. The crux of the problem lies in the decision of *when* to change the location of a component and decide where to deploy it in an adaptive manner. The Topology and Orchestration Standard for Cloud Applications (TOSCA) [11] is an xml-based standard to define the topology of cloud web services, their components, relationships, and the processes that manage them. It can also be used to define the orchestration of a mobile cloud application with an additional decision making

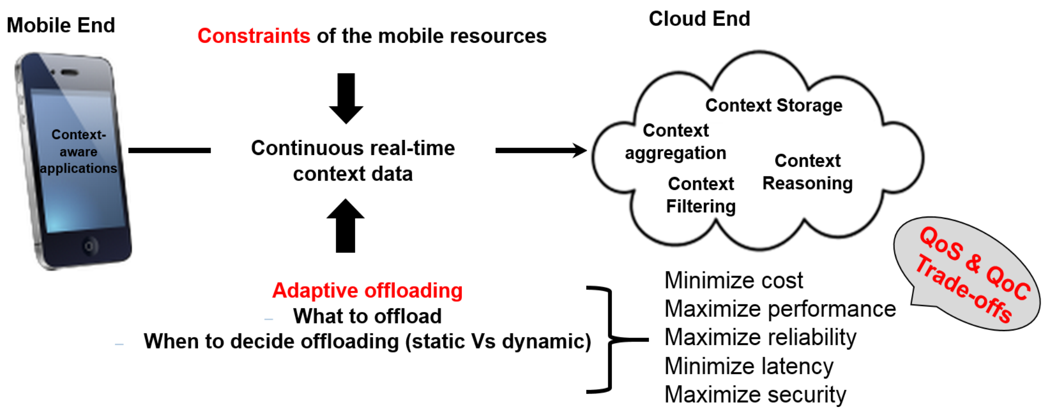


Figure 1. Overview of context-driven optimized federation in MCA.

support to choose the location of a component, based on the involved context factors and the resource-performance trade-offs. Abolfazli et al. [12] define decision making factors for the augmentation in MCC. The trade-offs between computations versus communication and performance versus latency are generally the most significant. User preferences with respect to privacy and security for sensitive data also play a role in the decision of which information to process where.

Computation on mobile devices always involves compromises and trade-offs [13] with respect to the required QoS [14] and QoC [6]. Context sensing and simple processing run on the mobile device and other computationally more expensive context management tasks like pattern detection, reasoning and learning are offloaded on the cloud infrastructure. The question is how we can effectively blend them to always achieve a feasible and beneficial augmentation? The constraints of a mobile environment play a role in the deployment decision. Additionally, The requirements of semantic knowledge and the resource characteristics of the application components make this decision highly dynamic. A primary challenge is to liberate device resources without compromising the QoS while preserving the QoC necessitating a Qo*-awareness in deployment decision making. Moreover, runtime uncertainty and erratic nature of the context information [15] periodically impact these deployment decisions and consequently performance; hence, the decision support needs to be flexible enough to ascertain the quality of its own decisions. The system should be aware of the impact of its decisions over time to optimize the runtime deployment and learn from its mistakes as humans do.

To address the needs of real-time dynamic systems, the Monitor-Analyze-Plan-Execute (MAPE-K) [16] framework with a concept of a feedback loop seems promising. We present a framework for self-adaptation of smart mobile applications as an instance of MAPE-K framework. Dynamic Decision Networks (DDNs), a specialized form of Bayesian Networks (BNs) are employed to analyze, plan, execute and update the knowledge-base at runtime with the changing context of the execution environment. Self-reinforcement [17] with the help of utility functions and temporal delays enables the framework to learn from its mistakes [18] in order to ascertain the quality of the taken adaptive decisions. Building on recent work of Bencomo et al. [19] in self-adaptive systems, we have adopted a model@Runtime approach applying DDNs, to incorporate the impact of trade-offs and to take an optimal decision under variability. DDN modelling is an emerging research topic, and researchers are investigating its use in the area of self-adaptation for autonomous systems in several domains [20][21][22][23].

In this work, we extend our previous work [1] and present a multi-objective application model with a fair trade-off between the required objectives. We highlight the challenges to exploit the cloud infrastructure opportunistically, and investigate the requirements to achieve a federation in MCA for smart mobile applications. Our presented framework is able to learn the deployment trade-offs of intelligent applications on the fly and is capable of learning from earlier deployment or configuration mistakes to better adapt to the settings at hand in a Qo*-aware fashion. To verify the effectiveness of our framework, a feasibility analysis is conducted on *Smart Lens*, an Augmented Reality (AR) based use case application. Research is conducted with respect to the communication cost and resource utilization when extending an AR application with context-awareness and cloud computing.

Our research offers the following contributions to support effective smart application deployment in a Qo*-aware federated environment:

- An investigation of the trade-offs that arise for data-intensive context-aware smart applications in MCA
- A federated dynamic deployment with respect to Qo* trade-offs
- Use of specialized probabilistic models to automatically learn the overhead of deployment trade-offs and compromises

This paper is structured in seven sections. In Section II, we give an overview of the background and the related work in MCA and discuss the gaps in state-of-the-art for federated deployments and decision making under uncertainty. Section III details our use case scenario motivating the need for smart deployment decisions. This is followed by Section IV, where we highlight the requirements and objectives of Qo*-aware decisions. Section V provides a brief account of our federated framework and the details about our approach of learning the trade-offs for dynamic deployments using a probabilistic decision model to mitigate the influence of runtime uncertainty. Finally, after evaluating our approach applied on our use case scenario in Section VI, the paper concludes and offers a discussion of topics of interest for future work.

TABLE I. The three offloading frameworks compared in terms of trade-off modeling and decision making.

Criteria	Features	MAUI	CloneCloud	ThinkAir	Our approach
Trade-Offs	Objective	Energy	Performance	Performance	Both
	Flexibility	Low	Medium	Medium	High
Decision	Context-Aware	High	Low	High	High
	Data store	Remote	Local	Local	Local
	Approach	Optimization	Optimization	Rules	Network
	Metric	Cost	Cost	Property	Property

II. Background and related work

In this section, we discuss and examine related work in the area of MCA, i.e., an active research domain in MCC. The strategies for runtime optimization in the presence of uncertain operational conditions are also briefed here.

A. *Mobile cloud augmentation (MCA) and Qo*-awareness*

Outsourcing the computation to the cloud can be beneficial to achieve an ideal QoS [14][24]. It has been proven to reduce the resource load on mobile devices [24], also for context-aware data-intensive applications [25]. This combination may result in improved energy efficiency and a reduced load on resources, such as CPU and memory. Beside context-awareness, a number of use cases combining data intensive mobile application and cloud computing have been described in the literature [26][27]. However, the use of context-awareness in cloud computing is often approached from a functional standpoint. The question for federated deployment 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 has enormous processing power, it is also possible to adopt a thin client approach and do almost all the work online. However, we cannot 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 and the distribution is never clear-cut in the scenario of intelligent systems considering trade-offs with respect to QoS and QoC [13]. Furthermore, acquiring knowledge from the available data for context information, requires a trustworthy mechanism where the ambiguity and uncertainty in context data can be mitigated.

Previous research [28][29] was carried out on how to realize platforms that allow the applications to make the partitioning decision while it is running, providing the user with the best experience possible. Even more impressive, some of these platforms can let applications enjoy the best of computation offloading by only making minor changes [28]. Context-awareness and computation offloading is added to achieve the desired functionality but the accompanying trade-offs regarding deciding *what* to offload and *when* are not explored. Narayanan et al. [30] predict the resource consumption on the basis of historical data by applications. They use this data to modify the fidelity of an application, based on the inputs and parameters received by any mobile application at runtime. Huerta and Lee [31] discuss a profiling based smart offloading policy using historic resource usage data. However, processing entire history logs is cumbersome. Cuckoo [32], ThinkAir [29] and MAUI [33] present an MCA model based on multi-objective criteria with respect to the performance and energy consumption. Cuckoo offers static offloading decisions without context-awareness. ThinkAir and MAUI make a decision based on the execution time, energy consumption, and previous execution history. MAUI processes the offloading requests by using the historic data to predict the execution time of any task without considering the input size of that execution, resulting in wrong prediction and offloading decisions [12]. We show a comparison of our approach with closely related approaches in Table I.

Chen et al. [35] investigate challenges related to the fact that each platform has its own capabilities and limitations to achieve certain QoS requirements. They present a context-aware resource management approach for service oriented applications with the ability to handle the inherent service and network dynamics and to provide end-to-end QoS in a secure way. QoC attributes and their modelling comes into play to capture the uncertainties in context data. Bellavista et al. [36] discuss the QoC requirements and their impact on context usage. Sheikh et al. [37] identify several quality indicators like precision, freshness, spatial resolution, temporal resolution and probability of correctness. The authors propose that these quality indicators are well-suited in ubiquitous systems for healthcare. However, no quantification mechanism has been proposed by the authors in order to evaluate the role of these parameters in critical decision support. Kim et al. [38] present the quality dimensions such as accuracy, completeness, representation consistency, access security and up-to-dateness for measuring QoC in ubiquitous environments.

Our solution starts without historic information and uses only the context at hand to predict the Qo*-aware dynamic deployment scheme for each component of a smart mobile application. Our solution does not focus on the partitioning scheme, but the optimal decision making for each of the component cloned on both the mobile and cloud ends.

B. *Decision support under changing circumstances*

Restating the obvious, intelligent applications have to take into account the runtime uncertainty in context data. Context sources are dynamic in nature. They can disappear and reappear 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. Pearl [39] explains the problem of uncertainty and argues that extensional or rule based systems cannot perform well under uncertainty. Probabilistic theory allows complex reasoning with a combination of observed evidences. Probabilistic systems can handle unseen situations addressing the influence of involved uncertainty. In our work, we are concerned with the mechanism of deciding when to reconfigure the deployment under a changing context at runtime.

Context-aware applications borrow decision models from artificial intelligence and machine learning field such as su-

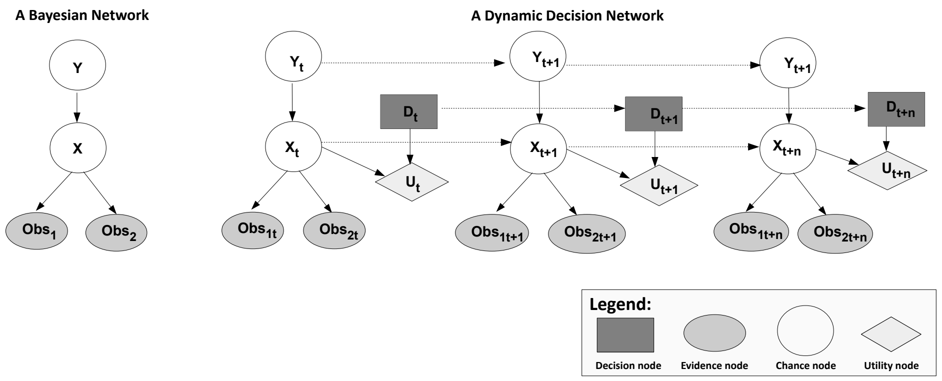


Figure 2. Structure of a DDN with dynamic chance nodes affecting utility nodes with decision nodes and evidences [34].

pervised learning (Bayesian models, decision trees, Markov models), unsupervised models (Neural networks), rules and fuzzy logic [25]. BNs are usually used to combine uncertain information from several sources to interpret high-level context information. Wolski et al. [40] presented the offloading decision as a Bayesian decision problem with a point of view of when to decide offloading under changing bandwidth using Bayesian theory arguing that in a Bayesian decision the inference of a new prediction is a well-defined function of the previously inferred prediction. Fenton and Neil [41] have used BNs for predictions of the satisfaction of non-functional aspects of a system. Esfahani et al. [42] 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. [43] using Markov models. Many works use utility functions to qualify and quantify the desirability of different adaptation alternatives. These works are QoS-based, applied in different domains for resource allocation [44], typically in component-based mobile and pervasive systems such as Odyssey [45] and QuA [46].

Bayesian based models are well researched in multi-criteria decision making [41][47] as well and generally applied in clinical artificial intelligence [48]. Nonetheless, researchers [19][49][50] have investigated the feasibility and tractability of DDNs to solve the problem of decision making under uncertainty in self-adaptive systems. Bencomo et al. [19] used DDNs to deal with the runtime uncertainty in self-adaptive systems. In recent years, two optimization techniques have been developed to address dynamic decision scenarios: partially observable Markov decision processes (POMDP) and dynamic decision networks (DDNs). Both techniques are powerful enough and aim at solving complex, real-life problems that rely on the postulates of multi-attribute utility theory and probability theory. Costa and Buede [50] present a comparison between both the approaches and conclude that POMDP lacks an ability to achieve any tractable model [51], while DDN systems can present feasible solutions for complex cases. The value structure can not be replicated in an explicit way in POMDP for a multi-optimal decision environment. The current state of the system has to be known, in order to use Markovian decision models. It involves an extensive re-engineering effort to the system since its value structure is implicit in its every state and transition. Although the basic dynamics of POMDP are still Markovian, as the current state is not directly accessible, decisions require keeping track of (possibly) the entire history of the process, contrary to the Markovian property where there is no need to keep a track of all the previous states and observations to take a decision or perform an action [52][53]. In a DDN, all nodes that contain value objectives are explicitly connected to a utility/value node.

C. *Learning the deployment trade-offs using DDNs*

Conditional probability distributions (CPDs) derived by analysing historical attribute values helped solve stochastic problems in the past. The 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. The adaptation module in our framework 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. We give a brief overview of the concepts of BNs and DDNs to understand the structure of the model used at runtime and how it is applied.

A BN is a Directed Acyclic Graph (DAG) that depends on Bayes' theorem [54] and CPDs. The graph is represented by a triplet (N, E, P), where N is the set of parameters, E is the set of arcs where each arrow declares the one parameter directly dependent on the one at the tail of the arrow, and P is the CPD for each parameter [34]. Figure 2 shows a BN with two chance nodes and two observation nodes. Chance nodes represent the influencing factors. BNs are able to reverse their inference logic due to the symmetry and usage of Bayes rule (given in Eq. (1)) and are able to update their beliefs on the fly as soon as a new evidence is observed [55]. Moreover, the Markov assumption (given in Eq. (2)) enables BNs and its dynamic counterparts, i.e., Dynamic Bayesian Networks (DBNs) to be fully operational even if an expert's opinion is fed to the model instead of an account of historic events.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

$$A^{t+1} \perp A^{(0:t-1)}|A^t \tag{2}$$

The computation of a posterior probability distribution over a model (or parameters) is called inference. It is one of the

TABLE II. Functional and non-functional (Qo*) requirements of the *Smart Lens* use case scenario.

Requirement	Description
Localization (QoS)	*Smart Lens* localizes the user within a building based on the camera. After recognizing a scene, it displays the position and orientation on a floor map and can give additional information about the user's surroundings.
Maximum Reliability (QoC)	The application should run reliably between the mobile and the cloud end without draining the mobile resources for other applications. A low reliability signifies that mobile is not the ideal execution environment, so the component should be deployed on the cloud.
Minimum Latency (QoS)	In MCC, the response time and performance of the application depends upon the location of the components and the amount of communication required to fulfil a task. Based on test data from CloneCloud [26], it should be reasonable to expect the application to return the results under 15 seconds.
Minimum Cost (QoC)	The switching cost between mobile and cloud impacts the performance of an application. It depends on the execution context, hence is considered a QoC requirement
Accuracy (QoS)	The results should be accurate. Because of the multitude of scenes that can be added to the dataset, it is also likely different locations will be represented by similar scenes, corridors and hallways.

basic operations offered by a BN. The precondition for inference is that the structure of the network/model is known and the prior probability distribution is already available. Learning can refer to the structure of the model, or the parameters, or both. Furthermore, learning may take place under either fully or partially observed variables. Learning offers a way to know the values of the parameters to properly explain the observed evidence. To represent parameters that change over time, it is possible to use a time-sliced network such that each time-slice corresponds to a time point in a form of a DBN. A BN/DBN does not necessarily require a historic data from one state to another for its inference, it can be suitable to perform the context reasoning for high-level context [56]. In order to realize proactive and situational decision making, two main concepts of Bayesian models are utilized, (1) Probabilistic QoS and QoC awareness using DBNs, and (2) dynamic decision making with DDNs to decide *when* to redeploy and *where*.

A DDN is a DBN that also includes a set of decision and utility nodes. The basic structure of a DDN is depicted in Figure 2. Decision node represents all the desired decision alternatives, connected to the utility node to compute its impact while making a decision. A chance node and other decision nodes can be the parents of a decision node. The 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 alternative given the evidence. A chance node, decision node and even utility node can be expressed as its parents. The decision alternative is chosen according to the Maximum Expected Utility (MEU) principle. If a decision parameter D contains decision alternatives $\{d_1, d_2, ..d_n\}$ and E is the evidence parameter containing $\{e_1, e_2, ..e_n\}$ for a state parameter with states $\{s_1, s_2, ..s_n\}$, then the expected utility based on Bernoulli's equation [57] for taking a decision alternative is give as:

$$EU(d_j) = \max_D \sum_i P(s_i \mid E, d_j) U(s_i, d_j) \tag{3}$$

We have employed DDNs to solve multi-objective, conflicting criteria problems while making Qo*-aware decisions over time for smart applications in MCC. Using a DDN is crucial in this research work for following reasons, given as:

- The type of relevant contexts evolve over time. Therefore, capturing the dependencies between the temporally variable relevance is difficult.
- If a static BN is employed, the interpretation of new evidence will lead to reinterpretation of previous evidence [58]. In order to overcome such a drawback, a DDN should be employed instead of a static BN.

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). This work explores and utilizes the trade-offs involved in combining a data-intensive mobile application with context-awareness and cloud computing, and investigates the deployment of such applications in federated MCC environment. After identifying the deployment and performance trade-offs for outsourcing data and computation, our approach addresses the federation concern by continuously learning and adapting under multiple conflicting QoS and QoC objectives.

In the next section, we explain a use case scenario of a smart application, motivating the federated deployment of its components with our Qo*-aware framework.

III. Motivating scenario for the dynamic deployment decision

Advances in mobile technology accelerate the use of high-end data-intensive platforms on mobile devices using built-in sensors, such as an AR platform using the camera of the device. Mobile AR allows the devices to recognise objects. It requires extensive processing for image recognition and matching.

A. Use case scenario

We considered an indoor positioning use case application called *Smart Lens* [59] to investigate the relevant trade-offs motivating the runtime deployment requirements for context-aware applications in MCC. Context-aware intelligence is often found in AR applications to help the user explore certain places, be it cities, expositions, 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. The core functionality of the application is to position the user based on the view of the camera. As soon as the application recognizes the scene, it displays additional information, such as a floor map, to give the user an idea of where he is as shown in Figure 3. Table II shows the functional and non-functional, i.e., Qo* requirements of the use case.

Smart Lens localizes the user within a building based on camera frames taken from a doorway. After recognizing a scene, it displays the position and orientation on a floor

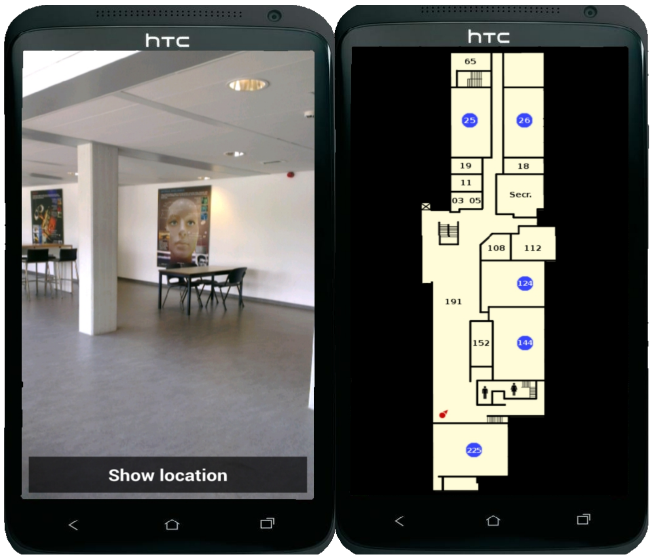


Figure 3. Scene recognition and image matching in the *Smart Lens*.

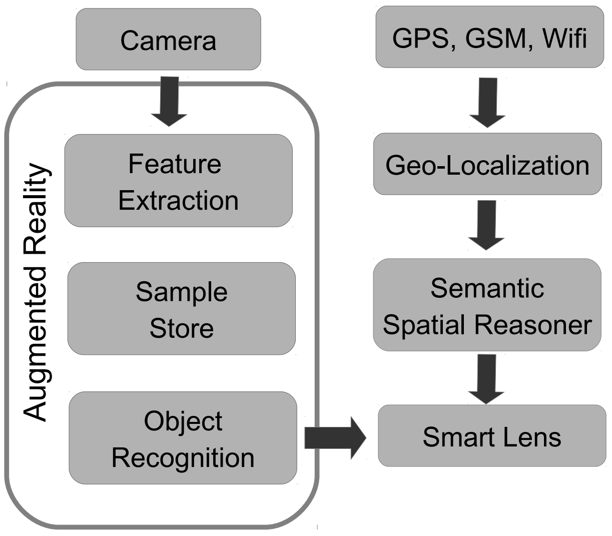


Figure 4. Overview of the components of *Smart Lens* use case.

map and can give additional information about the user's surroundings. The dataset can be large and continue to grow, adding more locations and poses, but can also be restricted to specific positions such as doorways to keep it practical. Modifiability can help the expansion of the dataset, which is having users capture and register their own scenes with the application. These users could be system operators, administrators of cooperating buildings, robots that explore the buildings or simply any user who wishes to register a location to navigate from or to. This application utilizes the location information of the user and the time of the day to further reduce the search space of objects to recognize and make the comparison smarter. We have analysed the performance and resource utilization trade-offs for our use case, motivating the need of the smart offloading decision. The next subsection discusses few of them.

B. Trade-off analysis

The AR components *Feature Extraction* and *Object Recognition* components require a relatively large amount of processing time, which not only drain the battery further but causes the application and the phone as a whole to slow down as well. Figure 4, provides an overview of the composition of the components for the *Smart Lens*. The resource and performance trade-offs are investigated for AR components by analysing a correlation between the latency of recognition, the structure of the dataset, i.e., its size and the quality of the images. Communication and computation trade-offs are also analysed for our use case.

The mobile device is held up facing an exact copy of a sample in the set. The use case application is started and shortly after, the time until recognition of the scene is printed on the screen. It is repeated for datasets of size 3, 50, and 100, consisting of high quality samples, low quality samples or a mixture of both. The size determines against what number of samples each scene needs to be compared, and the quality and detail in the samples themselves affect the amount of work needed for such comparisons.

Two different experiment scenarios are conducted: 1) when dataset is stored on the mobile device and the AR components perform the computations locally on the device, 2) when the cloud deployment for AR components is used keeping the entire dataset on the remote location as well.

1) Performance vs dataset size on the mobile: The results in Figure 5 demonstrate the effects of the dataset structure (on x-axis) on the recognition latency (on y-axis). Choosing the right dataset has an important influence on the performance. Larger local datasets slow down any data intensive scenario, in our case image recognition and AR based application requiring more resources impacting the performance negatively, and the use of high quality feature rich samples slows down detection when faced with an ideal scene.

However, it should be noted that smaller sets are less likely to contain the correct sample and require more computations if the applications is used for multiple detections, and lower quality samples have considerably more difficulties detecting scenes from a perspective that is not the same as the one when capturing the sample. Similarly, low quality samples also reduce the file size of the dataset. A smaller set not only reduces latency, calculation time and therefore energy consumption, it also reduces the file size, lowering the need for memory and possibly data communication. Smaller datasets can be obtained by filtering all samples in the system to those that are plausible given the current context of the device and the users. Additionally, devices with limited resources can choose to use datasets of lower quality, if offered, to save energy and calculations while sacrificing performance, hence instating the decision making on trade-offs.

2) Performance vs dataset size on the cloud: Figure 6 shows the latency when the dataset is placed on the cloud and the computation is done on the cloud. Lower latency with Wi-Fi shows that the performance of the application now improves with better Internet connectivity, as could be expected. It is not possible to offload fewer computations without fetching a part of the dataset, requiring additional communication with remote servers and anticipation and filtering of the dataset. AR applications using context-awareness to filter datasets, results in less memory usage and lower detection latency, whereas with cloud computing detection latency is approximately constant with respect to the dataset size, and no local memory is occupied

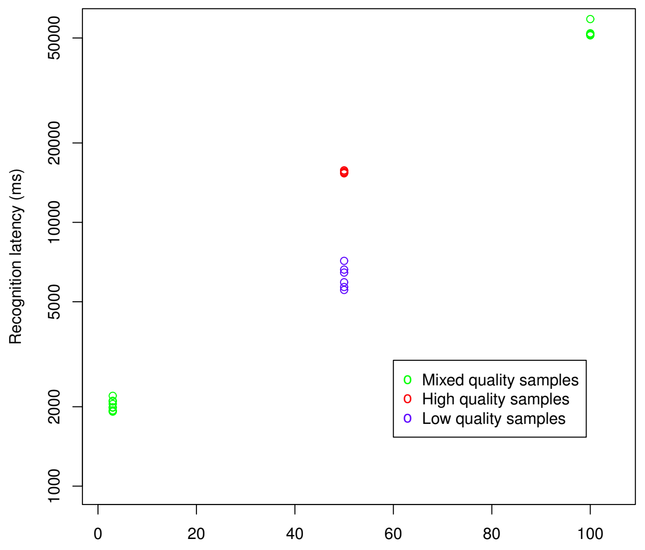


Figure 5. The latency in milliseconds (scaled logarithmically), when the recognition is performed on the mobile device with local dataset.

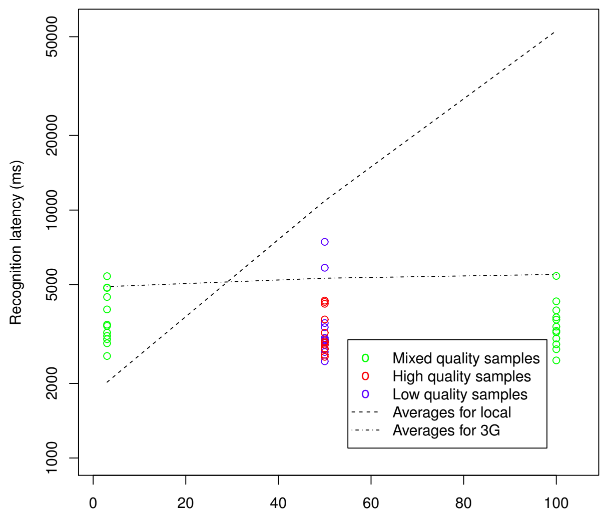


Figure 6. The latency in milliseconds with AR components and the dataset are offloaded to the cloud and accessed via WiFi. The trade-off curves of latency average for datasets stored on mobile and accessed via 3G are shown by the dashed lines.

by the dataset. But the latency due to connectivity type and the amount of data to be communicated is a major trade-off. The trade-off curves for our use are also shown in Figure 6. Placing the dataset on the cloud servers with the computation intensive components, leads to a need for a smart computation offloading method on the mobile device that learns the resource utilization and performance trade-offs in order to dynamically deploy the components between mobile and cloud infrastructure.

We will discuss the research requirements for automated decision support for the dynamic deployment considering the above mentioned trade-offs.

IV. Requirements for Qo*-aware smart deployment decision

A modular design philosophy for data and control processing on the mobile and cloud ends is needed to achieve a flexible distributed deployment. It simplifies redeployments and reconfigurations significantly. In our previous work [60], we demonstrated that a modular application design philosophy helps to support optimal mobile cloud application deployments. Moreover, we identified that many resource and performance trade-offs exist [13] in such a federated deployments for other use cases as well. The large number of parameters associated with the deployment configurations for these applications make it nearly impossible for developers to fine tune them manually. Therefore, automated deployment and optimization are necessary [61].

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. Furthermore, addressing the influence of runtime uncertainty in the context and its quality is an utmost essential task.

The investigated research requirements for a Qo*-aware deployment framework are given below:

1) **Offer reflective decision support:** The decision maker should be able to decide for which execution scenario the cloud is better and for which ones a cloud based deployment does not bring any added value, addressing *when* to offload in an automated way. Generally, the adaptation decision includes the ways to split responsibilities between devices, applications and the cloud servers. Since last decade, MCC research focus is to develop algorithms and techniques for deciding *whether* offloading would be beneficial or not and *what* to offload [62]. With continuous improvements in connectivity and mobile technology, the focus is shifting to smart offloading where the decision support is required with an aim of *when* to use MCA. Context-aware applications generate data at a large rate and sometimes in an uncontrollable fashion. The essence of the problem is finding the middle ground, determining what the mobile devices should handle and what the cloud should handle. Sending the raw data to the remote cloud servers is not always feasible [13]. Moreover, smart offloading demands a reflective decision support where the decision maker can look-ahead for the impact of the current decision and predict its impact on future situations.
2) **Process and provision the context:** As context is an essential constituent of smart applications. Runtime Context provisioning is inevitable in such a federated decision making. It always incurs a cost to profile the resources or other context parameters in mobile runtime environment. Furthermore, the continuous varying context and processing of the heterogeneous sensory data introduce challenges to take the most rational decision.
3) **Process the Quality-of-Context (QoC):** QoC raises more questions while dealing with automated adaptation. Context provisioning is a multi-level process [36] where low-level events are enriched through filtering and aggregation. For example, GPS coordinates are read from the sensor and translated into a high-level description of the location of the user. Finally, the desired high-level context is inferred

TABLE III. Objectives of the Qo*-aware offloading in MCC.

Objective	Description
Qo*-awareness	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. The system should be able to capture the real-time context of the user and his environment.
Self-Adaptive	The system must be self-adaptive at runtime to optimize the resource consumption of the application by outsourcing few components to the cloud.
Optimize Offloading	The system should detect the runtime context of the mobile device, i.e., CPU usage, memory consumption and battery usage. Runtime support to detect a change in QoC in a particular context type and optimize the offloading for Qo* requirements while performing opportunistic offloading. System should measure its impact on other context types before making any decision. Runtime support to detect a change in QoS before making any decision.
Resolve conflicts	The system should select the deployment strategy with respect to QoS & QoC requirements from the application's perspective in a total qo*-aware fashion, such as *Maximum Reliability*, *Minimum Latency* and *Minimum Cost* should be met in our use case.

through context reasoning. For instance, presence of the user can be inferred by his/her location. The adaptation decision is based upon this high-level context. End-to-end QoC control is a safeguard to monitor the quality of the context data throughout this multi-level process. The framework should consolidate the requirements of QoC in the context provisioning under varying context.

4) **Maintain the Quality-of-Service (QoS):** The re-configuration adaptation needs to be performed in an efficacious way without hindering the QoS of the applications. The possible re-configuration variants of any application can be determined at design-time in a static way but to achieve the multiple-objectives in MCA, it highly depends on the varying situation at runtime introducing the need of a decision support at runtime that does not hurt the QoS requirements of the smart applications in all possible deployment scenarios. This imposes a big challenge, especially for resource-limited mobile devices, when conflicting optimization objectives are involved. The decision maker should not only process the context but also predict the effect of the decision for future QoS requirements.

5) **Display retrospective behaviour for trade-offs in federated deployments:** Ascertaining the quality of the decisions is an important aspect in order to meet the QoS requirements that are directly affected by the context and its quality for smart applications. Context data is generated depending on the objectives of any smart application and its size varies accordingly. If the decision maker decides to achieve a certain requirement, obviously it has to compromise on another aspect. It is non-trivial for the decision maker to learn and memorize the decisions taken. Attaining a retrospective behaviour for MCC federated deployment decision is challenging due to the constrained mobile environment and the overhead has to be taken into account.

Table III shows the overall objectives of our framework according to the above mentioned requirements. An abstract overview of the QoC-aware adaptation is shown in Figure 7. It depicts the flow of end-to-end QoC and its processing at each level of the context-processing. The dotted area shows the significance of QoC-awareness in decision support using probabilistic models. We have used probabilistic models to mitigate the runtime uncertainty in the available context. In the next section, we will explain our Qo*-aware dynamic deployment framework and its learning mechanism using probabilistic models to achieve well-informed and reflective decision making with the above described features.

V. CONTEXT-DRIVEN DYNAMIC DEPLOYMENT APPROACH

Our framework consists of a loosely-coupled context-processing system along with an adaptation module. As shown in Figure 8, mobile hosts a *Dynamic Adaptation Module*, i.e.,

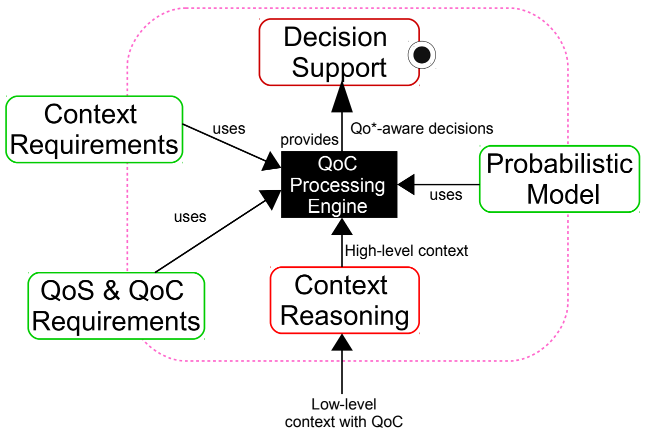


Figure 7. An abstract view of QoC-aware decision support in our framework.

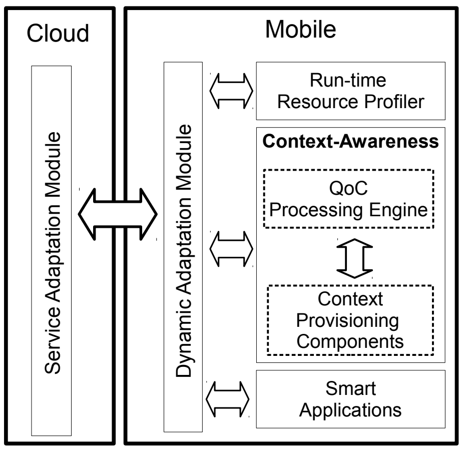

Cloud
Mobile
Service Adaptation Module
Dynamic Adaptation Module
Run-time Resource Profiler
Context-Awareness
QoC Processing Engine
Context Provisioning Components
Smart Applications

Figure 8. A blueprint of our federated framework and its dynamic deployment modules.

a component running on the mobile device to adapt the deployment configuration of the applications. 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. *Dynamic Adaptation Module* uses the model@Runtime approach and hosts the DDN model for adaptation. The next subsection details the working procedure of this module.

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 is able to decide on the opportunistic use of the cloud. Our decision making approach for redeployment and reconfiguration is explained in the steps mentioned below:

Information discovery and selection – The framework discovers and explores the application's runtime environment in order to get the context information to work with. Figure 9 shows a taxonomy of the runtime environment of a mobile device. Its resources can affect the ability to meet the QoS requirements and eventually, the decision of dynamic deployment. 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 the 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. The utility structure of the system is maintained here corresponding to the QoS requirements of *Minimum Latency* (L_{min}) and QoC requirement of *Maximum Reliability* (R_{max}) for the mobile device.

Analysis and decision making – The inferred context information is used to bootstrap the deployment decision or to change the configuration or behaviour of the application. System does a probabilistic analysis among the conflicting objectives in order to achieve the QoS and QoC requirements while making a decision. The computation or storage resources on the device and the communication cost affect the decision. For instance, the user points his mobile device on a scene. 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, scenes are then identified by matching the extracted feature points to those of known scenes 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 the video player, a context change is raised: the free memory on the hand-held increases.

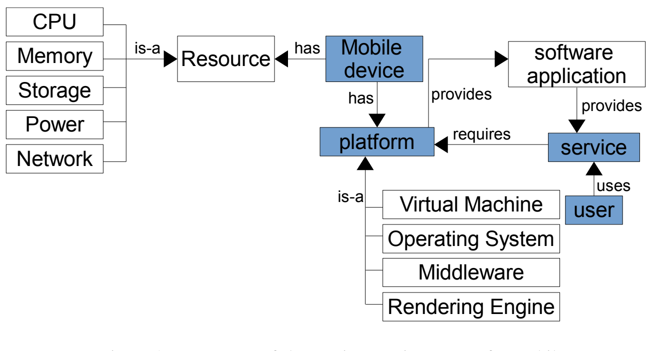


Figure 9. Taxonomy of the runtime environment of a mobile.

Enactment of the decisions – With a thin client configuration as in the previous step, the framework has to observe the real-time impact of the configuration to maintain QoS requirement of L_{min}. In order to decrease the 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. The 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, the 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 the future. The adaptation modules on both the ends communicates with the *QoC Processing Engine* for the deployment decision support. The context provisioning services use the *QoC Processing Engine* in order to provide Qo*-awareness.

B. Model structure

We present the details of the model structure and the involved multi-criteria parameters in this section. A DDN is modelled for the enactment of the decisions that change over time influenced by dynamic states and preferences. The first step is to identify the involved uncertain parameters and the causal relationships between those parameters [47]. Extensive interaction with the domain experts is vital to structure a quality model in order to fulfil the requirements. Table IV shows the identified parameters for our problem domain and their nature based on the requirements of the use case (see Section III). The value of the static variables is independent of their counterpart in multiple time slices. Dynamic parameters are affected in multiple time slices by their historic values. The effectiveness of reconfiguration decisions over time are investigated for multiple consecutive time slices. Each time slice contains an action taken by the system.

TABLE IV. Parameters types and values for QoC*-aware dynamic deployment.

Parameters	**Values**	**Nature**	**Availability**
CPU Usage	high, low	static	observed
Memory Usage	high, low	static	observed
Data Storage	remote, local	static	provided
Available Bandwidth	low, high	static	observed
Available Resources	yes, no	static	inferred
Required Connectivity	yes, no	static	inferred
Maximum Reliability	low, medium, high	dynamic	inferred
Minimum Switching cost	low, high	dynamic	inferred
Minimum Latency	minimum, average, maximum	dynamic	inferred

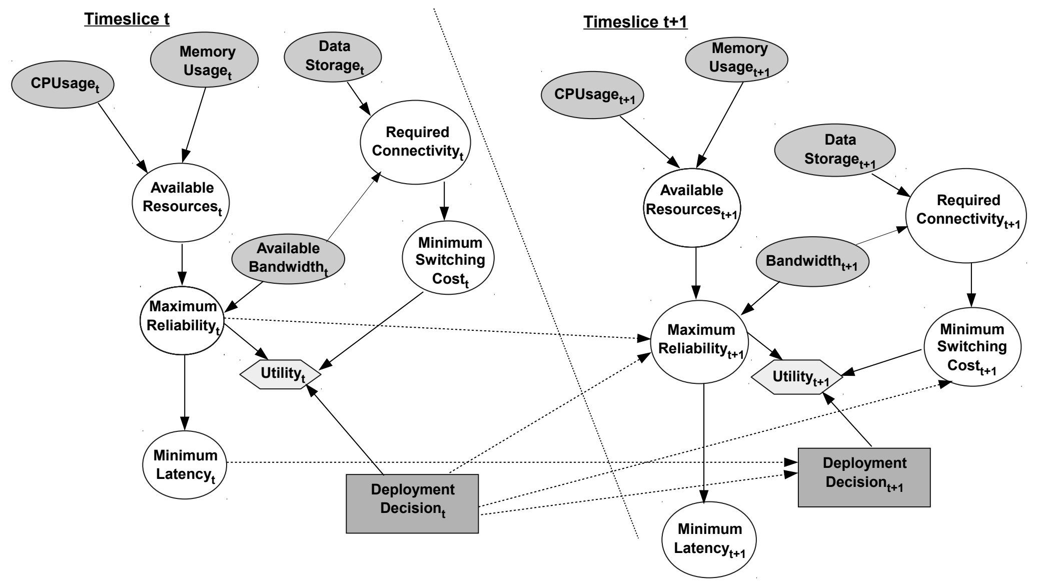


Figure 10. DDN model for dynamic deployment domain expanded in two time slices.

We have designed a DBN model to tackle the requirements 2, 3, and 4 identified in Section IV and a refined in the form of a two time sliced DDN model to address the rest of the requirements, i.e., 1 and 5 from Section IV. Figure 10 shows our DDN model.

The *Deployment Adaptation Module* uses this network to decide about the dynamic redeployments for the smart application components. We modelled the decision as a finite-horizon, sequential decision process [34]. At each time slice, the *Deployment Adaptation Module* decides on the fly about a component, whether to put it on the mobile or it should be running on the cloud. The time slice corresponds to a change in the context values of the execution environment of the mobile device depending upon the profiling interval. The use and feasibility of the DDN models are evaluated in [19][49] for other self-adaptive domains.

Our model expresses QoS and QoC requirements (*REQ*) by chance nodes and these requirements are causally linked by the involved context expressed as the observation nodes as shown in Figure 10. These chance nodes make a BN with the CPDs corresponding to the effects of *Deployment Decision* D_j alternatives {*mobile, cloud*} over conflicting *REQs* {*Maximum_Reliability, Minimum_Cost*} expressed as $P(REQ_i \mid d_j)$. *Available Resources* is a context parameter that stochastically varies according to the runtime environment parameters, i.e., *CPU usage* and *Memory* bringing uncertainty. *Bandwidth* is a random parameter observed dynamically. *Maximum Reliability* is the QoC parameter as it is inferred from the *Available Resources* on the mobile device and *Available Bandwidth* information, playing a vital role in decision making. Its value *low* shows that mobile device is not a reliable execution environment for the components, therefore, *Maximum Reliability* effects the utility of the decision. *Minimum Switching Cost* is another QoC parameter casually linked with *Data Storage* to capture the communication trade-offs while taking a *Deployment Decision*. The QoS parameter *Minimum Latency* is effected from the QoC parameter *Maximum Reliability* and plays a vital role in decision prediction, hence it is causally linked with the *Deployment Decision* in future time slice.

C. Utility function

A utility function computes the subjective choices of a decision maker for available decision alternatives and its outcomes. Elicitation of utility values requires the knowledge of the involved subjective probabilities [63]. In Bayesian inference, a decision is determined with both the utility function and the posterior probabilities. The utility values can be obtained by the domain experts or from the decision making preferences. We assigned the utility function for each of the preference criteria same as defined in [47]. We applied a linear transformation to normalize the utility values in order to reduce the computations for our DDN. The normalization formula is given as [47]:

$$U_i = 1 - \frac{V_{max} - V_i}{V_{max} - V_{min}} \tag{4}$$

The normalized utility values are given in Figure 11 for the most desirable decision alternative in the range from 1 to 100. For each REQ_i, the utility nodes express the utility function that takes the CPDs of the *REQs* and their priorities into account. $U(REQ_i \mid D_j)$ represents the numerical weight of each requirement and $P(REQ_i \mid E, D_j)$ represents the conditional probability for each *REQ* under the current observed context (evidence) *E* {*CPU Usage, Memory Usage, Available Bandwidth, Data Storage*}. The expected utility [34] for each decision is computed by Eq. (5) and the decision is chosen by the MEU principle.

$$EU(D_j \mid E) = P(REQ_i \mid E, D_j) U(REQ_i, D_j) \tag{5}$$

VI. Experimental Evaluation

In this section, we discuss the experimental setup and the results obtained towards an opportunistic offloading decision support using DDNs.

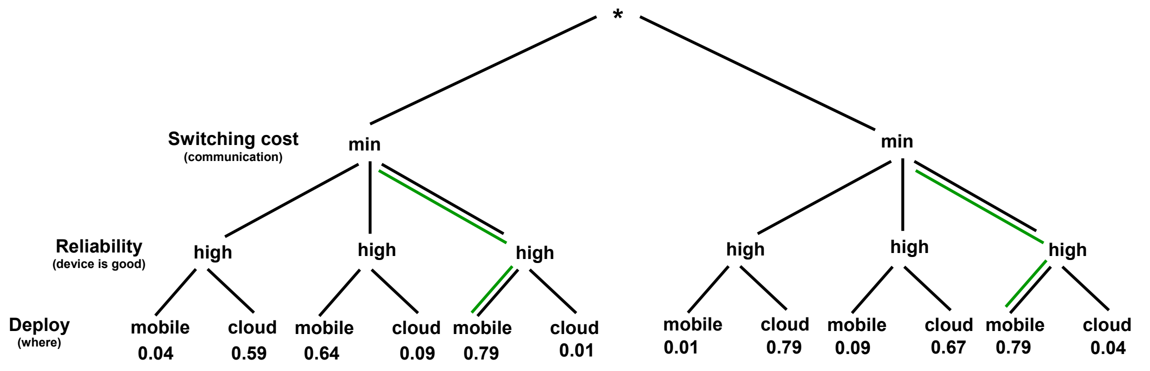


Figure 11. The numeric weights assigned to the conflicting objectives, the most favourable path is highlighted with a green line.

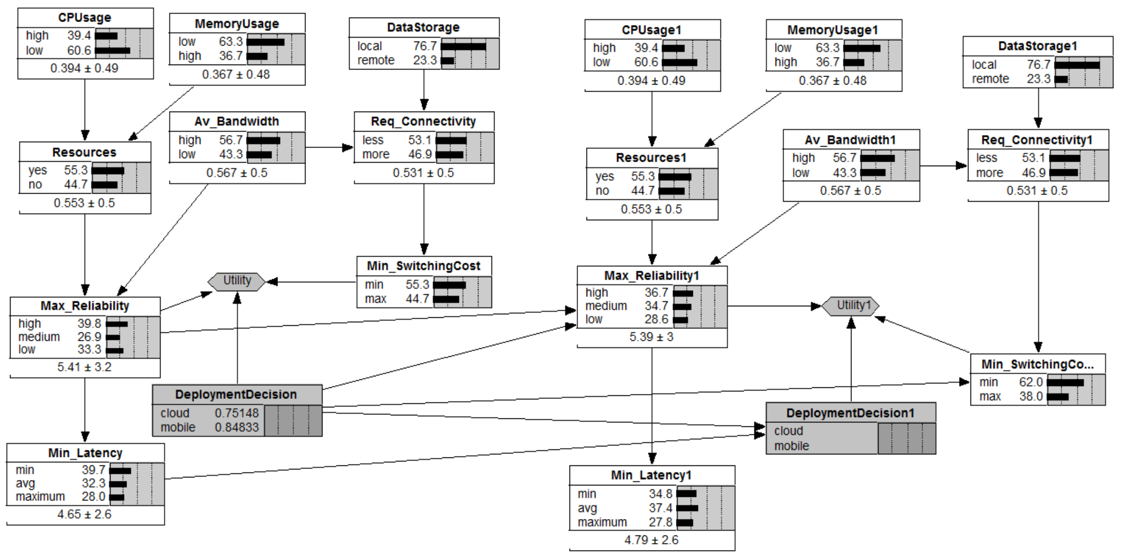


Figure 12. Prior probabilities of our DDN in two time slices.

A. *Enabling technologies & implementation*

In order to implement our framework, discussed in Section V-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 any scene.

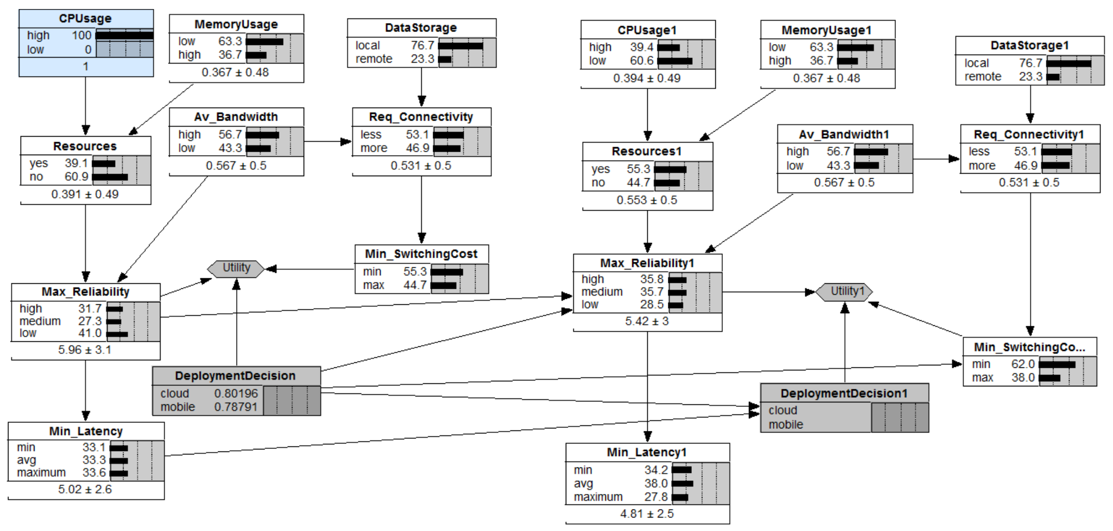
Our infrastructure runs VMware's open source Platform-as-a-Service (PaaS) offering known as Cloud Foundry on a server with 8 GB of memory and an Intel i5-2400 3.1 GHz running a 64-bit edition of Ubuntu Linux 12.04. A Java-based implementation has been used for *Runtime Resource Profiler* that captures the runtime context of mobile device. We have modelled a DDN-based probabilistic model for the components of our *Smart Lens* use case using the Netica development environment (http://www.norsys.com). We give a detailed account of the experiments conducted in Netica to analyse the deployment adaptation decisions in our model.

B. *Belief propagation and computation of the expected utilities*

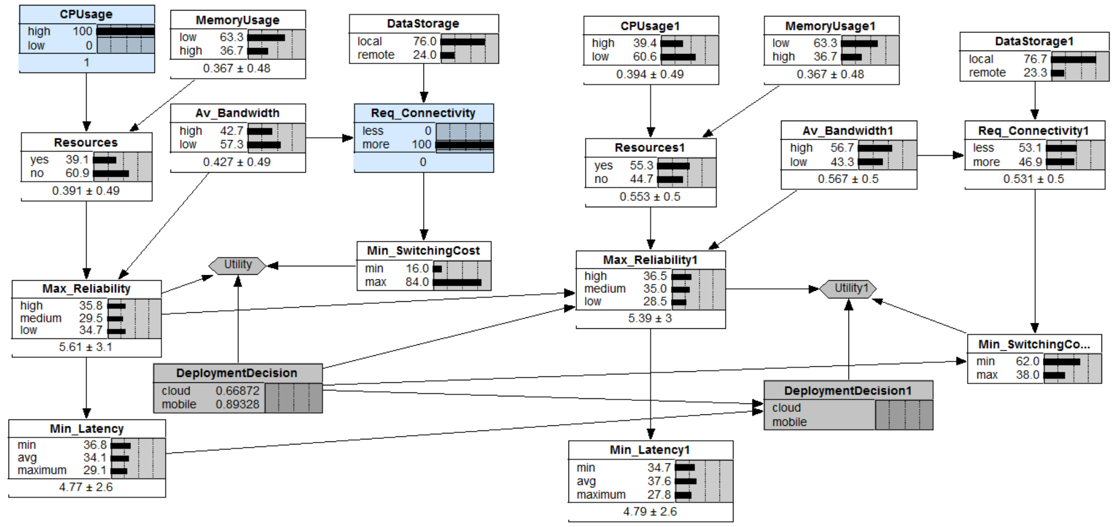
We investigated the belief propagation in our DDN model to analyse the decisions taken by it under changing context of the mobile device. The model can be bootstrapped with prior probabilities. These probabilities are learnt from experience dataset or can be set by domain experts in the same way as the policies can be set for rule based systems. Figure 12 depicts the initial computation of our model bootstrapped with the prior probabilities set according to the domain requirements. Under favourable conditions for execution environment, it can be seen that the decision to run the component on *mobile* has higher expected utility. As context is generated from mobile device, the prior probability for *Data Storage* is set to *local*. These prior probabilities are overwritten by the evidence in the form of observations from mobile resource profiling for belief propagation in the network. The experiments are conducted for the redeployment of *Object Recognition* component, whenever the execution environment changes on the mobile device.

In our first scenario, we observed the CPU usage as high with all the other parameters uncertain, the expected utility for the decision changes on the basis of the inferred *Maximum Reliability* as the *Switching Cost* remains *min* (see Figure 13a). When the *Switching Cost* changes to *max*, the expected utility is recalculated and it is triggered to choose *mobile* as a deployment option (see Figure 13b). This experiment shows that our model can effectively cope with the conflicting trade-offs and choose a favourable decision. In the second time slice, state of the Qo*-parameters are predicted and analysed. Once the decision is chosen it updates the decision for the future and recompute the belief propagation as shown in Figure 13c.

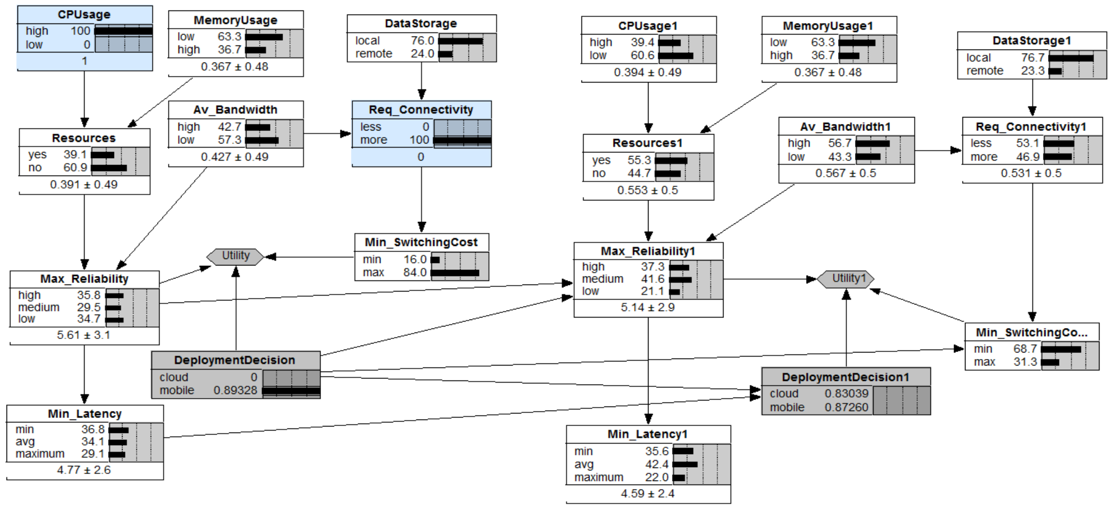
To evaluate the proactive adaptivity of our DDN model, we


CPUsage high 100 low 0 1
MemoryUsage low 63.3 high 36.7 0.367 ± 0.48
DataStorage local 76.7 remote 23.3
Req_Connectivity less 53.1 more 46.9 0.531 ± 0.5
Av_Bandwidth high 56.7 low 43.3 0.567 ± 0.5
Resources yes 39.1 no 60.9 0.391 ± 0.49
Min_SwitchingCost min 55.3 max 44.7
Utility
Max_Reliability high 31.7 medium 27.3 low 41.0 5.96 ± 3.1
DeploymentDecision cloud 0.80196 mobile 0.78791
Min_Latency min 33.1 avg 33.3 maximum 33.6 5.02 ± 2.6
CPUsage1 high 39.4 low 60.6 0.394 ± 0.49
MemoryUsage1 low 63.3 high 36.7 0.367 ± 0.48
DataStorage1 local 76.7 remote 23.3
Resources1 yes 55.3 no 44.7 0.553 ± 0.5
Av_Bandwidth1 high 56.7 low 43.3 0.567 ± 0.5
Req_Connectivity1 less 53.1 more 46.9 0.531 ± 0.5
Max_Reliability1 high 35.8 medium 35.7 low 28.5 5.42 ± 3
Utility1
Min_SwitchingCo... min 62.0 max 38.0
DeploymentDecision1 cloud mobile
Min_Latency1 min 34.2 avg 38.0 maximum 27.8 4.81 ± 2.5

(a) Expected utility with *CPU Usage* observed as *high*.


CPUsage high 100 low 0 1
MemoryUsage low 63.3 high 36.7 0.367 ± 0.48
DataStorage local 76.0 remote 24.0
Req_Connectivity less 0 more 100 0
Av_Bandwidth high 42.7 low 57.3 0.427 ± 0.49
Resources yes 39.1 no 60.9 0.391 ± 0.49
Min_SwitchingCost min 16.0 max 84.0
Utility
Max_Reliability high 35.8 medium 29.5 low 34.7 5.61 ± 3.1
DeploymentDecision cloud 0.66872 mobile 0.89328
Min_Latency min 36.8 avg 34.1 maximum 29.1 4.77 ± 2.6
CPUsage1 high 39.4 low 60.6 0.394 ± 0.49
MemoryUsage1 low 63.3 high 36.7 0.367 ± 0.48
DataStorage1 local 76.7 remote 23.3
Resources1 yes 55.3 no 44.7 0.553 ± 0.5
Av_Bandwidth1 high 56.7 low 43.3 0.567 ± 0.5
Req_Connectivity1 less 53.1 more 46.9 0.531 ± 0.5
Max_Reliability1 high 36.5 medium 35.0 low 28.5 5.39 ± 3
Utility1
Min_SwitchingCo... min 62.0 max 38.0
DeploymentDecision1 cloud mobile
Min_Latency1 min 34.7 avg 37.6 maximum 27.8 4.79 ± 2.6

(b) Expected utility with *CPU Usage* observed as *high* and *Switching Cost* observed as *max*.


CPUsage high 100 low 0 1
MemoryUsage low 63.3 high 36.7 0.367 ± 0.48
DataStorage local 76.0 remote 24.0
Req_Connectivity less 0 more 100 0
Av_Bandwidth high 42.7 low 57.3 0.427 ± 0.49
Resources yes 39.1 no 60.9 0.391 ± 0.49
Min_SwitchingCost min 16.0 max 84.0
Utility
Max_Reliability high 35.8 medium 29.5 low 34.7 5.61 ± 3.1
DeploymentDecision cloud 0 mobile 0.89328
Min_Latency min 36.8 avg 34.1 maximum 29.1 4.77 ± 2.6
CPUsage1 high 39.4 low 60.6 0.394 ± 0.49
MemoryUsage1 low 63.3 high 36.7 0.367 ± 0.48
DataStorage1 local 76.7 remote 23.3
Resources1 yes 55.3 no 44.7 0.553 ± 0.5
Av_Bandwidth1 high 56.7 low 43.3 0.567 ± 0.5
Req_Connectivity1 less 53.1 more 46.9 0.531 ± 0.5
Max_Reliability1 high 37.3 medium 41.6 low 21.1 5.14 ± 2.9
Utility1
Min_SwitchingCo... min 68.7 max 31.3
DeploymentDecision1 cloud 0.83039 mobile 0.87260
Min_Latency1 min 35.6 avg 42.4 maximum 22.0 4.59 ± 2.4

(c) Predicted state of the Qo*-parameters and beliefs in future.

Figure 13. Belief propagation in our DDN model for dynamic deployments on the fly.

TABLE V. Evaluation of our DDN model for different dynamic deployment

Expected utilities (EU)		Context parameters		Qo*-awareness parameters			Decision
Mobile	Cloud	Resource	Available bandwidth	Reliability on mobile	Switching cost	Performance	max(EU)
0.664	0.958	yes but with high CPU	high	high	min	avg	cloud
0.636	0.928	no	high	very high	min	avg	cloud
0.427	0.396	prior belief	prior belief	prior belief	prior belief	max	mobile
1.089	0.505	prior belief	low	low	max	max	mobile
1.006	0.563	no	high	high	max	avg	mobile

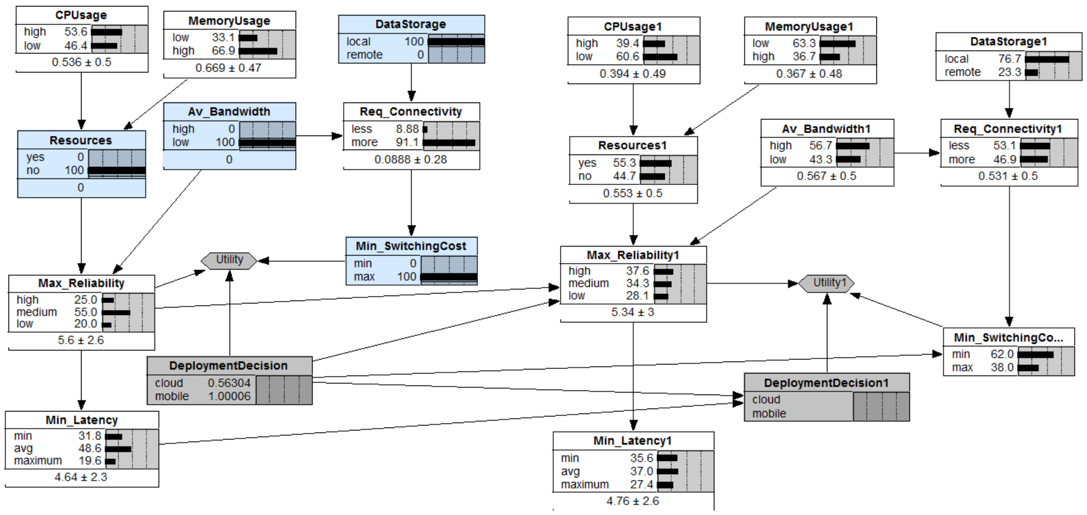
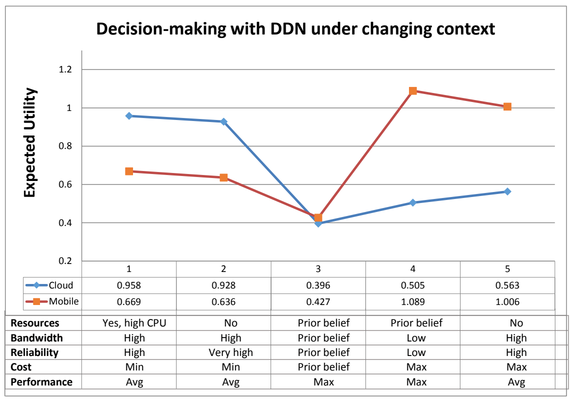


Figure 14. Expected utilities when there are no resources available on mobile device.

have conducted several experiments with the changing context environment on the mobile device. Table V shows the scenarios and the Maximum expected utility for these scenarios. Figure 14 depicts a snapshot of all the parameters and their beliefs. The *Maximum Latency* value is always analysed in the second time slice to achieve a proactive behaviour. In our first experiment, an automated trade-off analysis is done before choosing the cloud as the *Available Bandwidth* is high but *Resources* are on a low side. The decision model triggers an adaptation decision on the basis of the context parameters for mobile execution environment and do a trade-off analysis for Qo*-awareness. The most favourable decision alternative is chosen, as evident from the results in every case. Figure 15 shows the results for these experiments and chosen decision alternative for each runtime setting. The third scenario in the graph shows the intelligence of our model when there is no information available and it makes a decision in total uncertainty based on the prior beliefs.



	1	2	3	4	5
Cloud	0.958	0.928	0.396	0.505	0.563
Mobile	0.669	0.636	0.427	1.089	1.006
Resources	Yes, high CPU	No	Prior belief	Prior belief	No
Bandwidth	High	High	Prior belief	Low	High
Reliability	High	Very high	Prior belief	Low	High
Cost	Min	Min	Prior belief	Max	Max
Performance	Avg	Avg	Max	Max	Avg

Figure 15. Dynamic decision making for the changing context on the mobile device.

We have conducted several experiments with and without stressing the CPU, in order to evaluate the performance overhead of processing a DDN-based model for deployment decision making. The performance overhead of running a 2-sliced DDN on a Samsung S4 Android device is 5.8 milliseconds without stressing the CPU. But if the CPU is busy and stressed, the processing time goes up to 6 milliseconds. There is an overhead involved due to the evidence collection for runtime context. The processing with stressed CPU on the same device goes up to 16 milliseconds with an overhead of 10 milliseconds for the evidence collection.

VII. Conclusion and Future Work

MCA addresses the challenges to the resource limitation for mobile devices preserving the QoS requirements. Deployment for the components of smart applications in MCA is a non-trivial task in the presence of many resource-performance trade-offs and compromises. These compromises can affect the QoS or QoC for context-aware applications. The optimal strategy to deploy and configure intelligent applications with dynamic and heterogeneous resource availability cannot ignore the interplay between QoS and QoC. A 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 infrastructure, and to realize an opportunistic use of the cloud.

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 DBNs. 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 for future. Our first contribution is an analysis for the involved trade-offs for smart applications. To cope with the trade-offs for quality-aware deployment decisions, we present a Qo*-aware decision making framework based on DDNs in MCC domain. A feasibility analysis of incorporating DDNs for decision making was performed, 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 and 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 experiments have shown that an intelligent application can achieve optimal deployment for its components under a reasonable overhead, whenever the context is updated. However, the overall success of the model highly depends on the subjective probabilities and the utility function and its values. The sensitivity analysis [19][39] of these models validates their dependency on the prior beliefs and the utility values.

Applications of probabilistic theory and other artificial intelligence techniques can help to achieve the real meaning of smartness in several domains particularly in MCC. The limited tool support for their application is indeed a hurdle to widely utilize these techniques. We are actively working on our framework to realize the practical use of DDNs for dynamic deployment purposes in MCC using different available platforms for Bayesian inference and DDN support. In our work, we used two time slices network but we are interested to conduct a performance analysis of DDNs on a mobile device where it can be investigated that how many time slices are important in order to practically utilize these models in mobile environment without creating an overhead on device resources. 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 would be certainly very helpful as current tools support is fairly limited.

Acknowledgment

This research is partially funded by the Research Fund KU Leuven.

References

[1] N. Z. Naqvi, D. Preuveneers, and Y. Berbers, "Dynamic deployment and reconfiguration of intelligent mobile cloud applications using context-driven probabilistic models," in *INTELLI 2014, The Third International Conference on Intelligent Systems and Applications*, pp. 48–53, 2014.

[2] A. K. Dey, "Understanding and using context," *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[3] E. H. Aarts and S. Marzano, *The new everyday: Views on ambient intelligence*. 010 Publishers, 2003.

[4] 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, 1999.

[5] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[6] 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.

[7] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

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

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

[10] 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.

[11] OASIS, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0*, 2013 (accessed May 23, 2015). Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html.

[12] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 337–368, 2014.

[13] 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.

[14] 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].

[15] 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, 2010.

[16] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[17] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT Press, 1998.

[18] A. Filieri, M. Maggio, K. Angelopoulos, N. D'Ippolito, I. Gerostathopoulos, A. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, *et al.*, "Software engineering meets control theory," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2015.

[19] N. Bencomo, A. Belaggoun, and V. Issarny, "Dynamic decision networks for decision-making in self-adaptive systems: A case study," in *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '13, (Piscataway, NJ, USA), pp. 113–122, IEEE Press, 2013.

[20] C.-Y. Ting and S. Phon-Amnuaisuk, "Factors influencing the performance of dynamic decision network for inqpro," *Computers & Education*, vol. 52, no. 4, pp. 762–780, 2009.

[21] L. Portinale and D. Codetta-Raiteri, "Using dynamic decision networks and extended fault trees for autonomous fdir," in *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pp. 480–484, IEEE, 2011.

[22] A. Y. Tawfik and S. Khan, "Temporal relevance in dynamic decision networks with sparse evidence," *Applied Intelligence*, vol. 23, no. 2, pp. 87–96, 2005.

[23] T. Charitos and L. C. Van Der Gaag, "Sensitivity analysis for threshold decision making with dynamic networks," *arXiv preprint arXiv:1206.6818*, 2012.

[24] A. Khan, M. Othman, S. Madani, and S. Khan, "A survey of mobile cloud computing application models," *Communications Surveys Tutorials, IEEE*, vol. 16, pp. 393–413, First 2014.

[25] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, 2014.

[26] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, pp. 301–314, ACM, 2011.

[27] 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.

[28] 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.

[29] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Unleashing the power of mobile cloud computing using thinkair," *arXiv preprint arXiv:1105.3232*, 2011.

[30] D. Narayanan, J. Flinn, and M. Satyanarayanan, "Using history to improve mobile application adaptation," in *Mobile Computing Systems and Applications, 2000 Third IEEE Workshop on*, pp. 31–40, IEEE, 2000.

[31] G. Huerta-Canepa and D. Lee, "An adaptable application offloading scheme based on application behavior," in *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*, pp. 387–392, IEEE, 2008.

[32] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services*, pp. 59–79, Springer, 2012.

[33] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 49–62, ACM, 2010.

[34] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 74. Prentice hall Englewood Cliffs, 1995.

[35] S. Chen, J. J. Lukkien, and P. Verhoeven, "Context-aware resource management for secure end-to-end qos provision in service oriented applications," *Journal of Ambient Intelligence and Smart Environments*, vol. 3, no. 4, pp. 333–347, 2011.

[36] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A survey of context data distribution for mobile ubiquitous systems," *ACM Computing Surveys (CSUR)*, vol. 44, no. 4, p. 24, 2012.

[37] K. Sheikh, M. Wegdam, and M. v. Sinderen, "Quality-of-context and its use for protecting privacy in context aware systems," *Journal of Software*, vol. 3, no. 3, pp. 83–93, 2008.

[38] Y. Kim and K. Lee, "A quality measurement method of context information in ubiquitous environments," in *Hybrid Information Technology, 2006. ICHIT'06. International Conference on*, vol. 2, pp. 576–581, IEEE, 2006.

[39] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[40] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pp. 1–8, IEEE, 2008.

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

[42] 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*, p. 21, ACM, 2012.

[43] 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.

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

[45] 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.

[46] 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].

[47] W. Watthayu and Y. Peng, "A bayesian network based framework for multi-criteria decision making," in *Proceedings of the 17th international conference on multiple criteria decision analysis*, pp. 6–11, Citeseer, 2004.

[48] C. C. Bennett and T. W. Doub, "Temporal modeling in clinical artificial intelligence, decision-making, and cognitive computing: Empirical exploration of practical challenges," in *Proceedings of the 3rd SIAM Workshop on Data Mining for Medicine and Healthcare (DMMH). Philadelphia, PA, USA*, 2014.

[49] T. H. Bui, M. Poel, A. Nijholt, and J. Zwiers, "A tractable hybrid ddn–pomdp approach to affective dialogue modeling for probabilistic frame-based dialogue systems," *Natural Language Engineering*, vol. 15, no. 02, pp. 273–307, 2009.

[50] 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.

[51] T. H. Bui, M. Poel, A. Nijholt, and J. Zwiers, "A tractable hybrid ddnpomdp approach to affective dialogue modeling for probabilistic frame-based dialogue systems," *Natural Language Engineering*, vol. 15, pp. 273–307, 4 2009.

[52] A. R. Cassandra, *Exact and approximate algorithms for partially observable Markov decision processes*. Brown University, 1998.

[53] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra, "Solving pomdps by searching the space of finite policies," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 417–426, Morgan Kaufmann Publishers Inc., 1999.

[54] M. Bayes and M. Price, "An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs," *Philosophical Transactions (1683-1775)*, pp. 370–418, 1763.

[55] B. McCabe, S. M. AbouRizk, and R. Goebel, "Belief networks for construction performance diagnostics," *Journal of Computing in Civil Engineering*, vol. 12, no. 2, pp. 93–100, 1998.

[56] W. Dargie, "The role of probabilistic schemes in multisensor context-awareness," in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pp. 27–32, IEEE, 2007.

[57] D. Bernoulli, "Exposition of a new theory on the measurement of risk," *Econometrica: Journal of the Econometric Society*, pp. 23–36, 1954.

[58] R. Schäfer and T. Weyrath, "Assessing temporally variable user properties with dynamic bayesian networks," in *User Modeling*, pp. 377–388, Springer, 1997.

[59] Z. W. Bhatti, N. Z. Naqvi, A. Ramakrishnan, D. Preuveneers, and Y. Berbers, "Learning distributed deployment and configuration trade-offs for context-aware applications in intelligent environments," *Journal of Ambient Intelligence and Smart Environments*, vol. 6, no. 5, pp. 541–559, 2014.

[60] N. Z. Naqvi, D. Preuveneers, and Y. Berbers, "A quality-aware federated framework for smart mobile applications in the cloud," *Procedia Computer Science*, vol. 32, pp. 253–260, 2014.

[61] A. Ramakrishnan, S. N. Z. Naqvi, Z. W. Bhatti, D. Preuveneers, and Y. Berbers, "Learning deployment trade-offs for self-optimization of internet of things applications," in *Proceedings of the 10th International Conference on Autonomic Computing, ICAC 2013*, pp. 213–224, 2013.

[62] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[63] B. Houlding, *Sequential decision making with adaptive utility*. PhD thesis, Durham University, 2008.