Bayesian Network, and Probabilistic Ontology Driven Trust Model for SLA Management of Cloud Services

Obed Jules[1], Abdelhakim Hafid[1]
1Département Informatique et Recherche Opérationnelle
Université de Montréal
Montréal,Canada
{julesobe, ahafid}@iro.umontreal.ca

Mohamed Adel Serhani[2]
[2]College of Information Technology,
United Arab Emirates University
Al-Ain, United Arab Emirates
serhanim@uaeu.ac.ae

***Abstract---* To provide on-demand Cloud services, Cloud providers and customers are used to sign Service Level Agreement (SLA), which is an agreement on the level of service provided and its associated cost. Unfortunately, SLA is not as dynamic as the Cloud and the criteria on which a customer relies to select a provider can be of problematic nature. The provider must optimize its resources although he cannot predict the customer behavior that can make varying requests. This can have a direct impact on the availability of resources and thus may lead to SLA violations. To cope with this issue, we propose a framework that allows the customer to choose a trusted provider based on its reputation calculated using a Bayesian network and continuously updated via a Cloud directory. Moreover, we propose a smart and dynamic SLA scheme that uses a probabilistic ontology capable of detecting potential violations of contract parameters; in addition, it alerts the service provider about these violations. To evaluate our model we used data extracted from Amazon Web Service Elastic Cloud Compute and Google Compute Engine SLAs to simulate our proposal's environment. The results of our experiments show the effectiveness of probabilistic ontology for predicting SLA violation throughout some SLA parameters applied in Cloud environment.**

Keywords--- Cloud, Cloud Service Level Agreement, Trust, probabilistic Ontology, Bayesian networks, SLO, QoS.

I. INTRODUCTION

Cloud Computing is a novel computing paradigm that has experienced significant growth in the market place in recent years, in which a provider offers on demand IT services to customers through Internet. Cloud computing promotes strong collaboration between users and significantly increases business productivity. Cloud also offers enterprises a wide hosting space and storage for easy sharing with ubiquitous accessibility without prior investment in establishing the required physical infrastructure. All these and other features promote the popularity and the growth of Cloud Computing [1]. In the public Cloud, one of the three deployment models of Cloud Computing, a provider has the infrastructure and provides Cloud services through an SLA, which defines the nature of the service and the associated payment model. SLA [2] which is specified under the basis of negotiations between the provider and the customer, defines the role of the two antagonists for all the contract duration. If SLA requires the provider to efficiently manage its resources, it is for the customer a guarantee to receive certain quality of service (QoS) [3]. To measure the quality of the offered service, the customer relies on a specific element of SLA, which is Service Level Objective (SLO). SLO is a set of properties in which the provider indicates, for each quality requirement, what the customer should expect. The customer will then compare the received quality against the one promised in SLA; if discrepancies are detected, a contract violation is triggered. For example, Amazon EC2 [4] service (Elastic Compute Cloud) Web platform guarantees an annual availability of 99.95% [5]; however, in certain conditions the customer may not receive this availability due to unexpected events (e.g. service outage occurred in April 2011, [6]). In general, QoS assurance is not always guaranteed as it depends on external factors (e.g., customer limited bandwidth) and internal factors (e.g. poor SLA management and resources monitoring). In this paper, we analyze and propose a solution for the second category of risk factors that can affect QoS. Today the SLA management in Cloud environments faces several challenges including, and not limited to, the dynamic nature of monitoring SLA in Cloud, the complexity of determining the provider's trust level, and the difficulty of deciding about the selection criteria the customer might use to select a Cloud service provider. The customer might be supported in selecting the appropriate provider through a trusted third party that can evaluate and record the reputation of cloud providers.

We propose a framework that tries to address the above challenges through the implementation of three modules: (1) *Reputation-based Trust Module*: it evaluates and feeds trust ratings of each provider in a Cloud directory, based on its reputation defined as the probability to meet its commitments. We used a Bayesian model that uses a knowledge base, as prior information, to assess the provider's trust level. Thus, before starting SLA negotiations, the customer will have an idea on the provider's ability to meet its commitments; (2) *Intelligence Module*: this module uses an inference engine based on Multi-Entity Bayesian Network (MEBN); MEBN is a Bayesian first-order logic supporting a probabilistic ontology called PR-OWL [7]. This module estimates historical knowledge on resources availability; thus, given a complex parameter such as high availability, it determines by inference the likelihood of risks that the service might not be available; and (3) *Control Module:* it monitors and manages available resources and alerts providers in case of potential violations.

The remainder of this paper is organized as follows. Section II surveys existing contributions on SLA management. Section III introduces our framework and describes the role of their modules. Section IV describes the test environment and the implementation we have developed. Section V presents the evaluation scenarios we defined to evaluate our SLA management scheme, and the results we have obtained from the execution of these scenarios. Section VI concludes the paper and presents future work.

II. Background and Related work

Existing contributions that address QoS assurance in the Cloud can be classified into three categories: resources based management solutions, trust based models, and ontology based solutions.

A. *Resource Management*

Efficient resource management is key to improve the availability of services in the Cloud. Emeakaroha et al. [8] proposed a model, called LoM2HiS, that allows mapping low-level metrics (e.g., uptime and down time) to high-level SLA parameters (e.g., service availability). LoM2HiS is embedded into FoSII infrastructure [9], which monitors system parameters (e.g., resource availability and downtime) of the cloud infrastructure using a set of sensors. Then, it maps these parameters to high-level SLA parameters. Thus, it is able to detect potential SLA violations and notify the FoSII infrastructure upon notification, which can allocate more resources (e.g., to a VM thread) to avoid the occurrence of the violation. Even though these schemes (e.g., [8]) allow for efficient resource management, they (1) cannot learn, it merely recognizes and informs a violation but cannot predict violations based on historical data; and (2) do not compute the cost a customer needs to pay for additional resources.

Linlin et al. [9] propose an SLA framework based on dynamic resources allocation; such a framework allows managing resources in order to minimize the risk of SLA violations. It maps customer requirements (e.g., Response time) to infrastructure level parameters (e.g., CPU) while reducing the cost of resources allocations. This approach ensures that providers are able to manage the dynamic change of customers and handling heterogeneity of VM. Resources allocation had a cost; it is not evident that providers have to allocate resource for all type of customers without considering their priorities. In [10], the authors propose a classification of customers based on a price business model or affinities between the provider and customers. Then classify the customers by their priorities (e.g. Gold, Silver, and Bronze). This approach allows dynamic resources management and violations management according to the customer's priority, providing a high QoS to the preferential customers. However, it is not apparent how to establish a trust relationship between the provider and their customers.

B. *Trust Management*

There is no unique definition of trust; the closest definition to our context is given by [11] inspired by Gambetta: *"The trust of an entity C (Customer) has another entity P (Provider) to provide a Service (S). This is the probability that the entity P satisfies a request entity C for the Service S*". Trust is a key differentiator in Cloud environment; indeed, it allows customers to select cloud providers that likely will satisfy customer requirements. Trust [12] grants flexibility in negotiating SLA in a highly dynamic environments, such as Cloud environment. In addition, it promotes competition between providers. Unfortunately, the satisfaction of customers' requests cannot be guaranteed; this reduces customers' trust towards providers. The main limitation of existing trust management schemes (e.g., [13]) is that they do not consider uncertainty witch exists in cloud environments.

C. *Ontology Based Frameworks for SLA Management*

The dynamics of the Cloud environment requires SLA being able to self-adapt, being aware of its surrounding environment's changing parameters, and able to create new rules based on past experience. Hammadi et al. [13] present a framework for SLA implementation based on a trust model with reputation evaluated within a scale of [0, 5]. To incorporate intelligence in their SLA model, they used fuzzy logic inference engine that takes three input parameters: opinion recommendation, credibility and waiting time. Beom et al. [14] propose an ontology that manages tasks and resources on physical machines and job allocation to virtual machines in order to meet the requirements of SLA. They also, use OWL, which extends easily with new classes on demand. Both contributions [14, 15] are based on fuzzy models or deterministic languages; they do not allow reasoning to deal with uncertainty, which can be related to missing data, and compute the violations prediction probability using inference.

III. A framework for SLA enforcement in Cloud environment

In this section, we describe the components of our proposed framework (Fig.1.) for SLA management and enforcement in Cloud-based services. Our Framework consists of three modules: reputation-based trust module, intelligence module, and control module. The key contributions of the proposed framework include: (i) development of a smart SLA able to adapt automatically to the dynamic environment of Cloud; (ii) a trust model to rate trust level of a given cloud provider based on its reputation, in addition to the probability to honor its SLA commitment; and (iii) a novel scheme to detect potential SLA violations using probabilistic ontology based on Bayesian network reasoning.

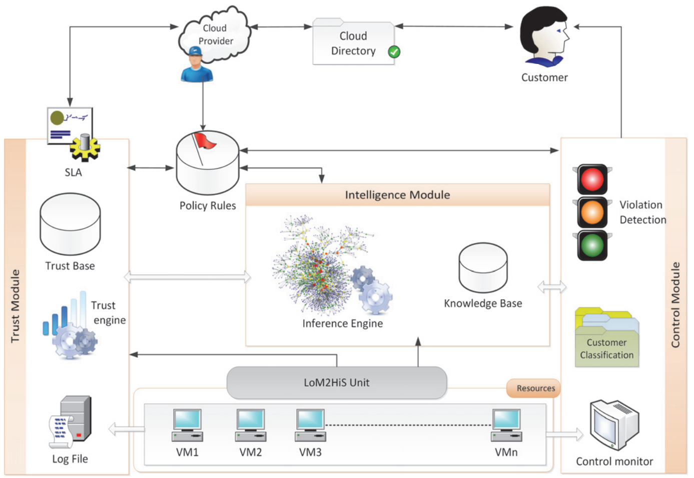


Fig. 1. Framework for SLA Management and Enforcement of Cloud-based Services

A. *Reputation-based Trust Module*

Given cloud providers are grouped in a Cloud directory [15]. In this paper, we classify/order cloud providers based on the probability (i.e., trust level) of each provider in violating the SLA; this probability is computed using a Bayesian inference engine that relies on historical data and measurements of low-level metrics. Bayesian rules allow calculating the posterior probability of any SLA violation event as hypothesis *(H)* based on a set of historical data *(D)*.

$$p(H \mid D) = \frac{p(D \mid H).p(H)}{p(D)} \qquad (1)$$

Where *p(H|D)* is the posterior probability of *H* given knowledge data *D*; *p(H)* is the prior probability for *H*; *p(D|H)* is the likelihood probability of *H* given *D*; and *p(D)* is the probability that would have happened whether or not *H* is true.

Reputation-based trust is generally calculated based on Customer's opinion evaluated by his perceived satisfaction; such an approach, in our opinion, is too subjective. In our trust model, we made use of an autonomous agent that assesses the reputation of a provider using quantitative parameters (e.g., available resources and saved session log files). To measure the trust level of a provider, two steps are necessary: (1) extracting the SLA parameters, which define the QoS offered by the provider to meet the expectations of its customers (e.g. CPU, storage, availability, response time). Then, map the low-level metrics (e.g. execution time) to high-level SLA parameters (e.g. availability), hence they are available and used for calculations. The LoM2HiS conducts two types of mapping (i) Simple mapping: a one-to-one correspondence (e.g., "disk space" is mapped to "storage" in SLA; and (ii) complex mapping: many-to-one mapping using a set of mapping rules as shown in Table 1.

TABLE I. COMPLEX CORRESPONDENCE PARAMETERS

Resources Metrics (Low-Level)	Parameters (High Level)	Mapping rules
downtime, uptime	Availability (Av)	$Av = 100\% * \frac{MTBF}{MTBF + MTTR.}$
inbytes, outbytes, packetsize, bandwidthIn, bandwidthOut	Response Time (Rtotal)	$R_{total} = R_{in} + R_{out}$ (ms)
mean time between failure(MTBF)	MTBF	$MTBF = \frac{\sum(Uptime - DownTime)}{Number\ of\ Fails}$

Once the mapping takes place, Bayesian calculations can be performed to assess the provider's trust level by comparing promised SLO described in the SLA, and the provided services whose characteristics are extracted from log files.

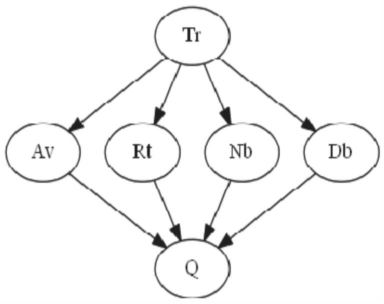
Figure 2, illustrates the naive Bayesian network that we have used to evaluate the trust level of a provider based on quantitative SLA parameters.

The naïve Bayesian structure supposes that observations are independents; this hypothesis can be overridden using an augmented naïve Bayes classifier called TANB (Tree Augmented Naïve Bayes). Formally, a Bayesian network is defined by: (i) an acyclic oriented graph noted **G**=(*V, E*) where *V* is a set of nodes, in our case ***V=(Tr, Av, Rt, Nb, Db, Q),*** and *E* is a set of arcs; (ii) a finite probability space (Ω, Z, *p*); and (iii) a set of random variables associated with nodes of the graph defined in (Ω, Z, *p*) as follows:

$$p(V_1, \ldots, V_n) = \sum_{i=1}^{n} p(V_i \mid C(V_i)) \qquad (2)$$

Where ***C(Vi)*** is the set of causes or parents of V_i in the graph ***G.***

Fig. 2. Provider's Trust naïve Bayesian Network



Variable	Description
Tr	Trust level
Av	Availability
Rt	Response time
Nb	Network bandwidth
Db	Disk bandwidth
Q	QoS

Intuitively a Bayesian network is represented as a causal graph where their properties facilitate inferences within this graph. Calculate inference values to estimate the graph's parameters can be done using some learning techniques. Generally, two methods are used to learn in a Bayesian network: learning parameters and structure learning.

B. *Parameters Learning*

A parameter in Bayesian network is the probability of observing a state value sought among the possible sates of a variable in all the historical data. For example, the overall trust level of a provider depends on the probability of its compliance with terms of each graph variable. This estimates the probability distributions or parameters relevancy compared to historical available data. Two approaches are usually used to learn parameters for the Bayesian network in order to estimate the probability distributions of variables. If data stored in the database is considered complete and the network structure is already known, parameters learning can be performed by statistical learning or by Bayesian learning.

1) Statistical Learning Approach

For a given set of parameters, denoted *(θ1…θn)* the statistical approach consists of calculating the frequency of occurrence of event θ_i, in the historical known data *D,* called Maximum Likelihood (ML).

$$\hat{P}(X_i = x_k \mid pa(X_i) = x_j) = \hat{\theta}^{ML}_{i,j,k} = \frac{N_{i,j,k}}{\sum_k N_{i,j,k}} \qquad (3)$$

Where *Ni,j,k* is the number of events recorded in the database for which variable *Xi* is in state *xk* and its parents are in state *xj* configuration. In practice, the database may have some missing parameters, which makes it impossible to estimate parameters by Maximum Likelihood. In this case, the Expectation-Maximization (EM) algorithm (see fig. 3) is applied to search in the network parameters until convergence to a local optimum of the expectation and maximization. Expectation estimates the missing *Ni,j,k* by calculating the conditional mean of the current network data. Maximization replaces the missing *Ni,j,k* by their mean values calculated by Expectation.

2) *Bayesian Learning Approach*

To predict Trust ***(T)*** of a provider, given data ***D*** based on parameter **θ**, the Bayesian prediction uses the Maximum a Posteriori (MAP) estimator to reflect the uncertainty around **θ** with a complete dataset observed by using prior parameters.

$$\begin{aligned} &\textit{Posterior} \;=\; \textit{Likelihood} \;.\; \textit{Prior} \\ &P(\theta|D) \;= p(D|\theta).p(\theta) = L(D|\theta).p(\theta) \end{aligned} \qquad (6)$$

$$\hat{P}(X_i = x_k \mid pa(X_i) = x_j) = \hat{\theta}^{MAP}_{i,j,k} = \frac{N_{i,j,k} + \alpha_{i,j,k} - 1}{\sum_k (N_{i,j,k} + \alpha_{i,j,k} - 1)} \qquad (7)$$

Where $\alpha\, i,j,k - 1$ are the Dirichlet distribution parameters associated with the prior. EM algorithm can also be applied to Bayesian learning, in case of incomplete observed data, to learn parameters by adding MAP given in Equation (5) in Figure 3. The later describes the main steps of Dempster EM algorithm.

$$\theta^{(t+1)}_{i,j,k} = \frac{N^*_{i,j,k} + \alpha_{i,j,k}}{\sum_k (N^*_{i,j,k} + \alpha_{i,j,k})}$$

```
Fix the Dirichlet prior
$X_v = \{X_v^{(l)}\} l = 1...N$   The observed dataset
$\theta^{(t)} = \{\theta^{(t)}_{i,j,k}\}$   The Bayesian network parameters at each iteration
Repeat:
    Expectation: parameters used to estimate missing data
    $N^*_{i,j,k} = E[N_{i,j,k}] = \sum_{i=1}^{n} p(X_i = x_k \mid pa(X_i) = x_j, X_v^l, \theta^{(t)})$   (4)
    Maximization: calculates new parameters by max likelihood
        $\theta^{(t+1)}_{i,j,k} = \frac{N^*_{i,j,k}}{\sum_k N^*_{i,j,k}}$   (5)
While $|\theta^{(t)} - \theta^{(t-1)}| \geq \varepsilon$
```

Fig. 3. Dempster EM Algorithm

3) *Structure Learning*

Once the best parameters are found, structure learning is used to find the network structure that reflects better the observed data. The structure learning used requests to maximize a score at each substructure and combine their results to find the model that best fits the observed data (**D**). The number of different structures for a Bayesian network with ***n*** variables is defined as follows:

$$r(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} = n^{2\partial(n)} \qquad (8)$$

A score ***S*** is said to be decomposable if it can be written as the sum or the product of functions that depend only of one vertex and its parents ***(pa)***. If ***n*** is the number of vertices in the graph, a decomposable score ***S*** must be the sum of local scores:

$$S(B) = \sum_{i=1}^{n} s(V_i, pa(V_i)) \qquad (9)$$

However, the structure learning approach illustrated in equation (8) increases exponentially the number of possible structures (ß), with the number of variables, making an exhaustive search infeasible in a large network space. To overcome this key limitation the Dirichlet Bayesian score (Score DB) in depicted in Equation (10), which is based on Equation (6).

$$scoreDB = p(\beta) \int_{\theta} L(D \mid \theta, \beta) p(\theta, \beta) d\theta \qquad (10)$$

To limit the search space, the K2 algorithm, a Cooper and Herskovits [16] heuristic, extend equation (10) by adding a prior on the structures as illustrated in the following equation.

$$SocoreDB(\beta, D) = \prod_{i=1}^{n} \prod_{j=1}^{qi} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \qquad (11)$$

The idea of the K2 algorithm is to maximize the probability of the structure (ß) given the data (***D***) to calculate the probability for multiple sub networks.

Parameters learning approach combined with finding the structure considering best parameters offers optimal way to predict the confidence level of providers even in the absence of data or missing parameters.

C. *Intelligence Module*

The intelligence module uses an inference engine able of using a knowledge base or observations to create new rules, inform the control module of potential violations, and provide updates on the provider's level of trust. The complexity of the Cloud does not allow predicting all possible rules to establish comprehensive database reasoning. Thus, an inference engine based on probabilistic ontology Bayesian networks is the most appropriate solution. Our intelligence module encompasses the following entities: (1) A knowledge base: it contains historical SLA violations including parameters, which are involved in each violation, and rules which were violated; and (2) an inference engine: it is based on Bayesian predictive model, thus, given a complex SLA parameter observed, such as availability, the inference engine predict the probability of future violation based on historical data.

1) *PR-OWL Ontology*

As OWL does not allow complex reasoning in cases where there may be non-identified/missing information or uncertainties, in this work, we use Probabilistic-OWL (PR-OWL) ontology, an extension of OWL that allows probabilistic reasoning based on Multi-Entity Bayesian Networks (MEBN). PR-OWL [19] is a representation of formal and explicit knowledge, which expresses the knowledge of an application domain. This includes (i) the entities and their relationships; (ii) the process and events triggered with those entities; (iii) statistical regularities that characterize the field; and (iv) inconclusive knowledge and uncertainties about all forms of knowledge.

OWL uses a conceptual formalism to represent domain entities in RDF (subject, object, predicate). PR-OWL uses combinations of classes and relationships linking these classes together. MEBN represents a given domain as entities that have attributes and are related to other entities. Knowledge about attributes of entities is encoded as a collection of MEBN fragments (MFrags). These fragments can be instantiated and combined to/from specific Bayesian networks for reasoning over complex situations. MEBN theory (MTheory) implicitly represents a joint probability distribution whose number of hypotheses can be unlimited. It uses Bayesian learning to refine the knowledge base calculated from observations.

The ontology composition using PR-OWL consists of the following steps (1) OWL ontology construction: it is viewed as XML information exploitable in OWL/XML format; (2) Formalization of uncertain relationships in MEBN logic: it represents the knowledge-base as a collection of MEBN fragments (MFrags) organized in MEBN Theories (MTheory); (3) A definition of conditional probability distribution (called Local Probability Distribution (LPD)): it is used to better define the probabilities of random variables; and (4) Creation of classes instances and their properties.

D. *Control Module*

The upstream implementation of this module relies on the work done by Mario Macias and Jordi Guitart in [10] to classify customers based on a business model. The control module performs monitoring and management of violations according to customer's priority (i.e., some customers have higher priority than others). It also notifies (a) low priority customers about the cause of SLA violation and the corresponding compensation where appropriate; and (b) Cloud providers of potential violations that are detected by the intelligence module.

LoM2HiS [8] allows mapping low-level metrics into high-level SLA parameters for SLA formulation and ontology construction. Fig. 4 shows the key interactions between mains entities of the proposed framework.

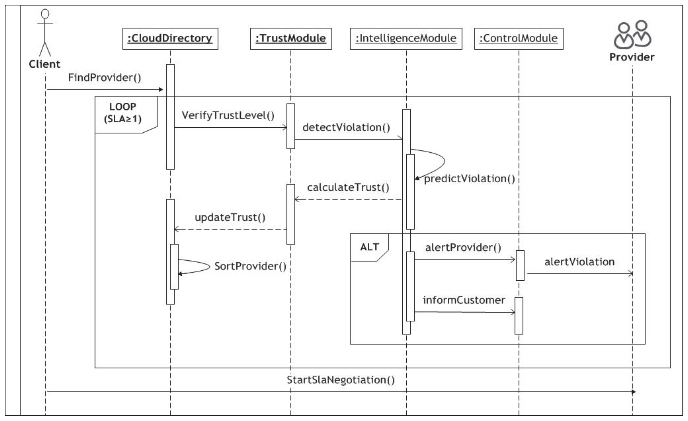


Fig. 4. UML sequence object interaction diagram

IV. Implementation

To evaluate our Framework we simulated our Reputation-based trust module and we implemented a MEBN probabilistic ontology for the Intelligence module. The information used for the experiment have been extracted from Amazon EC2, Google CE SLA [17] and their related pricing description [5, 18].

A. *Reputation-based Trust Module Implementation Description*

To implement the Reputation-based Trust Module we used BNT (Bayes Net Toolbox)[1] of Matlab [19]; BNT is a freeware available on GNU. The GPL license implements a large amount of algorithms and Bayesian functions. In the following, we describe the steps we have followed to implement our proposed trust module:

1. Create the network structure with the number of states that a variable can have. In our case all variables have two discrete states *High=2* or *Low=1*.
2. Use an appropriate algorithm to manage the inference engine. Then, we give evidence to some variables after observation for computing probabilities by inference.
3. Add some (Conditional Probability Distribution) to the variables from a knowledge file using a random process with parameters learning. In fact, for missing parameters in our case, we have used EM algorithm. Unfortunately, the EM algorithm does not support continuous observations. To overcome this issue we have used discretization for continuous variables.
4. Score the best structure for better solution is obtained using the K2 BNT algorithm implementing the ScoreDB equation (8). As the EM algorithm, the K2 algorithm works only with discrete variables and without missing parameters in the dataset.
5. Compute marginal probability with the evidence variables and the inference engine given in order to evaluate predictions and their likelihood.

B. *Probabilistic Ontology based PR-OWL*

To build the ontology, (1) we used Protegé toolbox [24]; it is a java based software available under open source license; it provides a GUI interface for writing OWL2 ontologies in electronic OWL/XML format; and (2) we imported ontology in Unbbayes[2]; it is a java open source plugin able to support PR-OWL2 format to build probabilistic ontology over MEBN Framework and make Bayesian inference reasoning on OWL ontology. To reflect "realistic" ontology, we used data extracted from Amazon EC2 and Google CE SLAs (see Table II)

TABLE II. DATA *S*AMPLE EXTRACTED FROM AMAZON EC2 AND GOOGLE CE SLAS

	Amazon EC2			**Google CE**			
SLA Parameters	***Availability***		***Penalty***	***Availability***			***Penalty***
	99.95 % >			99.95 % >			
	99.95 % - <99.0 %		10%	99.00% - < 99.95%			10%
	< 99.0		30%	95.00% - < 99.00%			25%
				< 95.00%			50%
Services Price	***Type***	***Memory***	***Price/hr***	***Type***	***Cores***	***Memory***	***Price/h***
	m3.large	7.5	$0.140	n1std-4	4	26GB	$0.280
	r3.xlarge	30.5	$0.350	Highcpu4	4	3.60GB	$0.176
	i2.xlarge	30.5	$0.853	g1-small	1	1.70GB	$0.853

The PR-OWL version 2.0 offers a great compatibility with OWL. Thus, we can easily work with Protegé tools to build a basic OWL ontology, and then export the OWL/XML file into Unbbayes tools. PR-OWL 2.0 an OWL extension, builds a bridge allowing communication between deterministic ontology and probabilistic ontology.

[1] *https://code.google.com/p/bnt/*

[2] *http://sourceforge.net/projects/unbbayes/*

In fact, the OWL ontology is composted by SLA parameters with classes that define the contract properties including an identifier, penalty parameters and SLO characteristics. The SLO includes Quantitative Property and Qualtative Property. The quantitative Property class includes the SLA quantitative properties like response time, VM parameters, data performance, network performance, the service availability and the metric class which describes how the parameters are evaluated (like “<”.”>”,”=”,”boolean”,…).

In our experiment, we have used some OWL data properties that capture the information type that can support OWL classes. In our case, some data properties are boolean like, *hasLowResponseTimeReport* capture boolean information (*yes or no*). Unbbayes uses Resident node to map relation bettwen the OWL data properties to PR-OWL. The resident nodes use as input parameter the OWL classes, like VM, responsetime, availability values. Then, we have computed the posterior probabilisties to assess the causal effects between nodes.

V. Evaluation

In this section, we describe the evaluation of our proposed framework in particular the trust module and the probabilistic ontology. The main objective is to access the possibility of predicting the provider level of trust while maintaining a high accuracy. Then use probabilistic ontology to predict SLA terms violations.

A. Evaluation Environment Setup

The following describes the main steps we have built to experiment our trust module:

1. Build the network with variables order: *Trust (Tr=1); Availability (Av=2); Response time (Rt=3); Network bandwidth (Nb=4); Disk bandwith (Db=5); QoS(Q=6),* as shown in Fig. 2.
2. Format dataset using a random process implemented with Matlab. We produced a training dataset of 750 records. Then, we used a third of the training set: 250 records for testing. Three components were produced: (i) A naïve network with complete data where all the variables are discrete; (ii) Hid 20 percent parameters on the previous network in order to analyze and compare the results; (iii) Created a mixed network containing discrete and continuous variables.
3. Then, we conducted learning of network parameters with complete data and then, with missing data using EM algorithm, and compute the Maximum likelihood parameter estimation with maximum 15 EM iteration.
4. We conducted structures learning, using K2 algorithm, in order to produce a better subnet structures that fit the problem solution.

Our Probabilistic ontology provides an efficient way to anticipate violations given a prior as knowledge. The main queries submitted to assess our ontology used evidences to answer properly these two questions:

1. Query 1: Based on the historical data, what is the probability that a violation has occurred?

   Evidence: *Verify that the service availability report is low.*

2. Query 2: What is the impact of response time on the availability parameter.

   Evidence: *Verify the response time report is low.*

VI. Results and Discussion

In the first experiment, we tested our proposed solution on three networks (mixed, naïve discrete, and EM) and for each scenario, we calculated the marginal probability *P(Tr=High|Av, Rt, Nb, Db, Q)*. The results are illustrated in the Receiver Operating Characteristic (ROC) confusion matrix presented in fig. 5. The figure shows the percentage of good prediction and its corresponding recorded false alarm rate. Ideally, the more the prediction is closer to 100%, the more false alarm rate is closer to 0%. In our case, the EM algorithm gives a good prediction percentage 82%, however, mixed network gives a prediction ratio of 98% with 40% of false alarm rate. When the false alarm rate is consistent it is necessary to study other parameters before taking a final decision. The good performance of the EM algorithm can be explained by the fact that this algorithm replaces missing parameters by the average expectation of the network. If several missing parameters are found below the expectation, so the EM algorithm will provide a better performance than the other approaches.

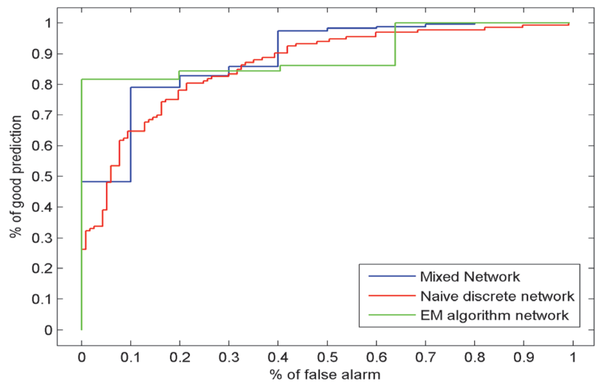


Fig. 5. ROC of confusion matrix from parameters learning

In the second experiment, we use K2 algorithm to learn Bayes Network Structure. Fig.7, shows a clear increase of the correct prediction rate compared to figure 5. This is due to the learning network structure using K2 algorithm. Naïve discrete network provides a prediction rate of 95% along with 20% of false alarm rate. Therefore, this prediction rate exhibited by our module seems to be perfect given the small dataset used. But it might not reflect the same results in a coherent knowledge experimental database.

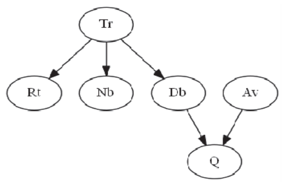


Fig. 6. Structure Learning on Complete Data by K2 Algorithm

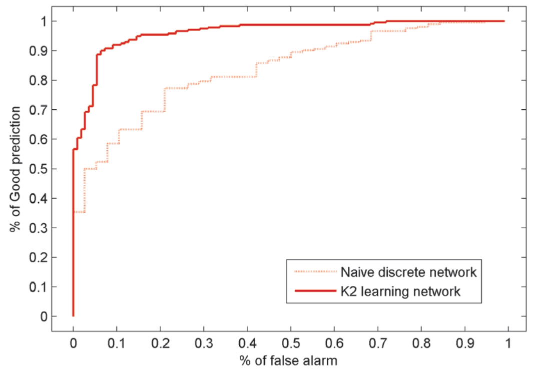
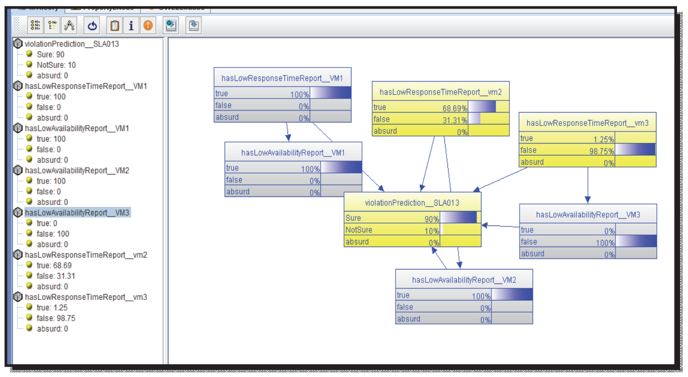


Fig. 7. ROC of Confusion Matrix of Learned Structure by K2 Algorithm

We present in the rest of this evaluation section the results of our probabilistic ontology as depicted in figure 8. It shows on the left side the queries and the evidences, while on the right side shows the results and the causal effects that upgrade or downgrade the probability of an event based on evidence. It illustrates using an example of a given SLA reference *(SLA013)* how the response time (*hasLowResponseTimeReport_VM1*) affects the availability of the virtual machines *VM1, VM2* and VM3 *(hasLowAvailabilityReport_)*. Thus, prediction can be held with a violation of a percentage certainty, in our example it is around 90%.



```
violationPrediction__SLA013
  Sure: 90
  NotSure: 10
  absurd: 0
hasLowResponseTimeReport__VM1
  true: 100
  false: 0
  absurd: 0
hasLowAvailabilityReport__VM1
  true: 100
  false: 0
  absurd: 0
hasLowAvailabilityReport__VM2
  true: 100
  false: 0
  absurd: 0
hasLowAvailabilityReport__VM3
  true: 0
  false: 100
  absurd: 0
hasLowResponseTimeReport__vm2
  true: 68.69
  false: 31.31
  absurd: 0
hasLowResponseTimeReport__vm3
  true: 1.25
  false: 98.75
  absurd: 0
```

hasLowResponseTimeReport__VM1
hasLowResponseTimeReport__vm2
hasLowResponseTimeReport__vm3
hasLowAvailabilityReport__VM1
violationPrediction__SLA013
hasLowAvailabilityReport__VM3
hasLowAvailabilityReport__VM2

Fig. 8. Unbbayes Window of the Probabilistic Ontology Result

However, the application of our approach in a real cloud environment requires sharing of log files. Among others, it can also pose confidentiality and security problems.

VII. Conclusion and Future Work

Managing SLA in Cloud ecosystem is as complex as the dynamic nature of the Cloud and its evolving services. Selecting the appropriate cloud providers should heavily rely on evident data such as the provider's trust, and its ability to guarantee the pre-agreed SLA. In this paper, we proposed a framework that uses a Bayesian network to measure the provider's reputation and continuously update it if violations of SLAs have been detected. We also, proposed a dynamic SLA management scheme that relied on probabilistic ontology to detect violations of SLA's parameters and notify concerned providers about the triggered violations. We evaluated our model using four network structures, and the results we have obtained showed that the K2 learning network provides the best prediction that reached 100%. Therefore, Bayesian network was proven to be an effective approach to predict provider's trust level as to prevent such violations while considering the uncertainty of cloud service environment. As future work, we are planning to extend the implementation of the proposed solution and experiment it on a real environment.

References

[1] G. Motta, N. Sfondrini, and D. Sacco, "Cloud computing: a business and economical perspective," in *2012 International Joint Conference on Service Sciences (IJCSS 2012), 24-26 May 2012*, Los Alamitos, CA, USA, 2012, pp. 18-22.

[2] E. Casalicchio and L. Silvestri, "Mechanisms for SLA provisioning in cloud-based service providers," *Computer Networks,* vol. 57, pp. 795-810, Feb 2013.

[3] S. Kona, A. Bansal, M. B. Blake, S. Bleul, and T. Weise, "WSC-2009: A Quality of Service-Oriented Web Services Challenge," in *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, 2009, pp. 487-490.

[4] *Amazon Elastic Compute Cloud (Amazon EC2)*. Available: https://aws.amazon.com/ec2/

[5] Amazon. (2013). *Amazon EC2 Service Level Agreement*. Available: http://aws.amazon.com/ec2-sla/

[6] S. Hagen, M. Seibold, and A. Kemper, "Efficient verification of IT change operations or: How we could have prevented Amazon's cloud outage," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012, pp. 368-376.

[7] PR-OWL. *A Bayesian extension to the OWL Ontology Language*. Available: http://www.pr-owl.org/

[8] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, 2010, pp. 48-54.

[9] W. Linlin, S. K. Garg, and R. Buyya, "SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011), 23-26 May 2011*, Los Alamitos, CA, USA, 2011, pp. 195-204.

[10] M. Macias and J. Guitart, "Client Classification Policies for SLA Negotiation and Allocation in Shared Cloud Datacenters," in *Economics of Grids, Clouds, Systems, and Services. 8th International Workshop, GECON 2011, 5 Dec. 2011*, Berlin, Germany, 2012, pp. 90-104.

[11] N. Chung Tien, O. Camp, and S. Loiseau, "A Bayesian network based trust model for improving collaboration in mobile ad hoc networks," in *Research, Innovation and Vision for the Future, 2007 IEEE International Conference on*, 2007, pp. 144-151.

[12] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems,* vol. 43, pp. 618-644, 2007.

[13] A. M. Hammadi and O. Hussain, "A Framework for SLA Assurance in Cloud Computing," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 393-398.

[14] M. Yong Beom, J. Sung Ho, and L. Jong Sik, "Ontology-Based Resource Management for Cloud Computing," in *Intelligent Information and Database Systems. Third International Conference, ACIIDS 2011, 20-22 April 2011*, Berlin, Germany, 2011, pp. 343-52.

[15] A. Hammadi, O. K. Hussain, T. Dillon, and F. K. Hussain, "A framework for SLA management in cloud computing for informed decision making," pp. 1-17, 2012.

[16] R. W. Robinson, "Counting unlabeled acyclic digraphs," in *Combinatorial mathematics V*, ed: Springer, 1977, pp. 28-43.

[17] Google. *Google Compute Engine Service Level Agreement (SLA)*. Available: https://developers.google.com/compute/sla

[18] Google. *Compute Engine*. Available: https://cloud.google.com/products/compute-engine/

[19] "MATLAB R2013a and BNT Toolbox ", ed. Natick, Massachusetts, United States.: The MathWorks, Inc.