SLA learning from past failures, a Multi-Armed Bandit approach

Lise Rodier, David Auger, Johanne Cohen
PRiSM-CNRS,
45 avenue des Etats-Unis,
78000 Versailles, France
Email: {David.Auger, Johanne.Cohen, Lise.Rodier}@prism.uvsq.fr

Hélia Pouyllau
Thales Research and Technologies,
Campus de Polytechnique, 1 avenue Augustin Fresnel,
91767 Palaiseau Cedex
Email: helia.pouyllau@thalesgroup.com

Abstract—**A Service Level Agreement (SLA) is a contract between a customer and a Network Service Provider (NSP) defining services to one or more destinations at a given Quality of Service (QoS). Once committed, the SLA can be violated without the customer being able to predict that.**

We focus on offer selection mechanisms according to QoS and taking into account past SLAs' violations. We propose an algorithm, using a minimizing-regret technique, which provides an estimation of the reliability of given NSPs to the customer. Our approach only requires an end-to-end monitoring tool for used paths and depends on the customer's history.

I. INTRODUCTION

In the Internet network, more and more applications and services require some level of Quality of Service (QoS). We focus on a customer requiring access to a destination prefix with QoS guarantees. The customer has to choose among several contract offers proposed by his neighboring Network Service Providers (NSPs). The Service Level Agreement (SLA) is a contract which specifies the required QoS guarantees and charging conditions between a provider and a customer. The customer is not necessarily an end-user and may be an Internet Access, a provider, an NSP, an enterprise etc. Once the customer has selected an offer, an SLA is established between the chosen NSP and the customer.

We focus on the customer's offer selection according to QoS and taking into account the violation of the constraints encountered in the past with an NSP. The objective of the customer is to learn the reliability of given NSPs in order to establish an SLA having a low probability of being violated. Whoever the customer is (an end-user, a provider, an NSP, an enterprise etc.), this objective still holds. For instance, the endusers' satisfaction strongly relates to destination availability; the interest of a provider is to ensure the reliability of the path he will propose in his own offers.
The customer has to choose an offer having satisfying QoS as well as a competitive price. If the offered QoS and the price are the only parameters affecting the customer's payoff, this choice is simple and corresponds to the offer maximizing a trade-off QoS/price. However, the customer's payoff does not depend only on these two parameters. Indeed, once the customer has chosen an NSP's offer and established an SLA with him, the latter could be violated and the customer unsatisfied. In this context, the choice is more complex than one may think. In order to make a good choice, the customer must be able to estimate the reliability of NSPs. For that purpose, the customer learns from past SLAs' failures using a learning algorithm.

Nevertheless NSPs are competing with each other; their common goal is to be selected by the customer. The complexity of the NSP's choice lies in the fact that the NSP has to make an offer which is competitive with those of the other NSPs without any knowledge of the said offers. Otherwise the NSP is not selected and does not gain anything. Furthermore, the NSP needs to manage its remaining capacity because it affects its failure rate. The NSPs' behavior generated by this competition has an impact on the customer's behavior.

The SLA negotiation problem is described in Section II. The next section details the Multi-Armed Bandit problem and the algorithm used by the customer. Section IV presents the NSPs' selection algorithms for the offers. Finally, in Section V, the results of simulation performed on a network example are discussed.

II. SLA FAILURE DETECTION MODEL

In the model described below, we focus on one customer connected to the Internet through n Network Service Providers. This system is illustrated by Figure 1, the customer is linked to three NSPs, NSP 1, NSP 2 and NSP 3.

The customer request is defined by QoS requirements, a duration and a destination prefix. The customer makes such a request to his neighboring NSPs. In return, the NSPs make their own QoS offers with a price depending on their current states (i.e. of their resources, of their targeted market, etc. any indicator qualifying the request value). Note that, we do not address the bandwidth reservation problem [1], [2], we consider that each NSP has a fixed initial capacity which decreases during the establishment of an SLA. Hence, in our modeling, the state of an NSP is based on its remaining capacity, such as defined in [3].

Each of those NSPs has one or more disjointed path to the destination which may have different specifications such as QoS and *violation rates*. Furthermore, each NSP has a set of internal paths that meet QoS capabilities vectors. The offer made to the customer is the combination of the set of paths and all internal available QoS capabilities. NSPs are allowed not making any offer because they cannot reach the prefix through their own connection, or not at the requested QoS. We assume that they do not motivate their absence of offer proposal.

The customer's goal is to select an offer, and so an NSP, with a good trade-off QoS/price defined as his payoff. The customer chooses an offer among the ones proposed by his neighboring NSPs granting him a good payoff and establishes an SLA for the requested duration.

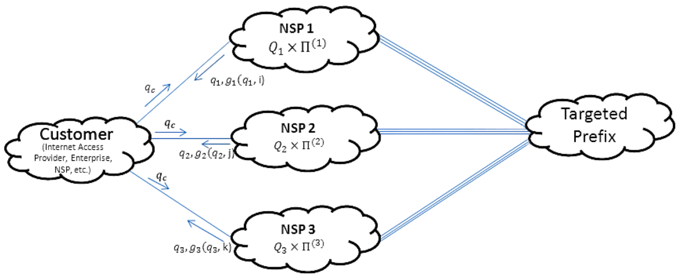


Fig. 1. Example: A customer network is connected to 3 NSPs. Each NSP i has a defined set of QoS offers, Q_k, and a set of disjointed paths to reach a targeted prefix, $\Pi^{(k)}$. When the customer sends a request denoted q_c, each NSP k might make an offer proposal q_k at a price $g_k(.)$.

A. The SLA negotiation scenario

a) The customer request: The QoS request q_c of customer c is a triplet $q_c = (b_c, d_c, l_c)$ which components are bandwidth (b_c), delay (d_c) and loss-rate (l_c) required guarantees. Moreover, request q_c has a requested duration Δ_c and a destination prefix. The parameter Δ_c corresponds to the duration of the SLA once established.

Without a loss of generality, we consider that a customer is linked to a unique type of request, different QoS values leading to different customers. Thus a customer is *characterized by his QoS attributes.*

b) NSPs answer: Each NSP k chooses an internal path and a QoS capability from its set of QoS capabilities Q_k. Furthermore, NSP k make an offer to the customer by choosing QoS capability in a set $Q'_k \subseteq Q_k$ which satisfy the customer's requested QoS guarantees. Section IV discusses the algorithms to determine such policies.

An offer is composed by a supported QoS capability $q_k \in Q'_k$, $q_k = (b_k, d_k, l_k)$ for NSP k and a price $g_k(q_k, i)$.

The equation (1) is an example of a cost function for the NSP's offer price. As with all cost functions used in this model, the price of the QoS offer is calculated from the ratio $\frac{\text{bandwidth}}{\text{delay} \times \text{loss-rate}}$ multiplied by the price bound of the one applying it. With the bandwidth being a capacity per time, the delay a time unit, and the loss-rate without unit, this ratio provides an estimate of the capacity needed. The cost function of the customer, the NSPs and also the NSPs' external paths are made from this function model.

The price $g_k(q_k, i)$ calculated from the QoS capability q_k and the path i associated with this QoS capability.

$$g_k(q_k, i) = p_k \times \frac{b_k}{l_k \cdot d_k} + g_i^{(k)}(q_k) \tag{1}$$

where p_k is the NSP margin and $g_i(q_k)$ the cost function of path i for QoS q_k. The offered price depends on the cost function of path i which is $g_i(q_k) = p_i \times \frac{b_k}{l_k * d_k}$ with p_i being the cost of the path. This takes into account the price that the NSP has to pay in order to reserve path i for requested QoS q_k.

If the customer accepts the NSP's offer then path i is reserved and the capacity of the NSP is decremented by b_k, the bandwidth offered. Every time a customer sends a request, each neighboring NSP could make him an offer according to the algorithm mentioned above. In case the QoS bandwidth asked by the customer exceeds the NSP's capacity, it does not make an offer. This assumption implies that the NPSs cannot cheat and propose an offer they are not able to satisfy.

c) The customer choice: If the NSPs do not make offers that do not support the QoS required by the customer, in the same way the customer does not accept an offer whose price is higher than his budget. The budget is computed using the customer c price bound p_c and the cost function g_c applied to the asked QoS q_c. An offer having a price higher than $g_c(q_c)$ is rejected. Considering the payoff of the customer for such an offer it is equal to 0.

The customer chooses between the remaining offers and perceive a reward. We define the customer's reward as being The customer's reward measures his satisfaction and is defined by :

$$rew_c(k, q_k) = \begin{cases} 1 - \frac{1}{1+\beta}\left(\beta P_{c,k} + Q_{c,k}\right) \\ 0 \text{ in case of failure} \end{cases} \tag{2}$$

with $P_{c,k} = \frac{g_k(q_k,i)}{g_c(q_c)}$ for the price, $Q_{c,k} = \frac{1}{3} \times (\frac{b_c}{b_k} + \frac{l_k}{l_c} + \frac{d_k}{d_c})$ for the QoS and $\beta \in 0, 1$ the price sensibility in the reward.

The reward takes into account the ratio between the price of the selected offer and the price the customer could afford to pay for the QoS he asked represented by $P_{c,k}$, hence $0 \leq P_{c,k} \leq 1$. The higher the price of the offer is, the more the ratio is close to zero. On the contrary, the more the price is close to the price bound of the customer, the more the ratio tends to 1. This ratio $P_{c,k}$ is preceded by a factor β that gives more impact to the QoS than the price in the reward.
The second ratio $Q_{c,k}$ is a normalization of the sum of each component, in the requested QoS and the offered QoS, ratios. Each term $\frac{b_c}{b_k}$, $\frac{l_k}{l_c}$ and $\frac{d_k}{d_c}$ equals 1 when the QoS offered is the same than the QoS asked and is close to 0 when the QoS offered is higher. Therefore $0 \leq Q_{c,k} \leq 1$, it tends to 0 when the QoS offered is better than the asked one and to 1 in the opposite case. This formula does not take only one aspect of the offer into account, either a cheap price or a better QoS, but a combination of both. Indeed a small price indicates a QoS offer close to the request and vice versa, this reward provides a balance between these two parameters.

We normalize the sum $\beta P_{c,k} + Q_{c,k}$ and obtain $\frac{1}{1+\beta}\left(\beta P_{c,k} + Q_{c,k}\right) = 1$ when the SLA offer is exactly the same as the request and tend to 0 if better.

B. The SLA failure

Our model focuses on the SLA error detection. The failure of an SLA comes from either the NSP, based on its remaining capacity, or the path selected. Failure based on path does not depend on the NSP and has a fixed probability of failure. For the failure rate based on remaining capacity, the failure rate increases gradually as the NSP's remaining capacity decreases. This is due both by an overload on the equipments and by the fact that in case of failure on an equipment less alternative paths are available. We borrow from [3] a failure function, denoted $f(.)$, which depends on the NSP current capacity :

- 0% of the capacity : $f = 1$
- 0% to 5% of the capacity : $f = 0.95$
- 5% to 10% of the capacity : $f = 0.8$
- 10% to 20% of the capacity : $f = 0.6$
- 20% to 30% of the capacity : $f = 0.2$
- more than 40% of the capacity: $f = 0.01$

We consider both cases, capacity and path violation or path violation only. The customer's goal is to maximize his average payoff and learn the reliability of his NSPs in order to establish an SLA having a low probability of being violated. For that purpose we explore a minimizing regret algorithm.

III. Customer's offer selection algorithm using past SLAs failures

The Multi-Armed Bandit problem: The customer's selection problem could be seen as a Multi-Armed Bandit (MAB) problem. In a multi-armed bandit problem, at each decision epoch, a player chooses one action (or arm), and receives a reward from it. The reward is drawn from a fixed probability distribution given the arm. We consider that an arm corresponds to one NSP. Other approaches may be considered, such as an arm for each offer of an NSP, nevertheless it would imply to know each NSP's set of offers. Therefore, we establish that an NSP proposes only one offer to the customer at each time and this NSP corresponds to one arm in the MAB problem.

We will make an assumption that the reward (payoff) corresponding to all SLAs of the NSP is drawn from a *fixed probability distribution*. This assumption seems to be verified by simulation results.

A. The Multi-Armed Bandit problem

In the MAB problem, a player faces K independent gambling machines with an unknown stationary probability distribution for reward. At each round, the player chooses one action, and receives a reward from it. There are K arms and each arm i has a probability distribution P_i with the expected value μ_i. For $t = 1, \dots, T$, at round t, the player chooses an arm among the set of arms $\{1, \dots, K\}$ from the past (actions and observations). At the end of the T rounds, the player is evaluated in terms of the difference between the mean reward of the optimal arm and the mean reward. Precisely, this loss is due to the fact that at each round the policy does not always select the optimal machine. The regret after T steps on K machines is :

$$R_T = \mu^* \times T - \sum_{j=1}^{K} \mu_j \times T_j \tag{3}$$

where μ^* is the expected mean of the optimal machine, μ_j the expectation of machine j and T_j the number of times machine j was played. The regret for a fixed policy is the expected loss after a fixed number of rounds T.

This type of problem symbolizes the dilemma between exploration and exploitation. The player's objective is to obtain a maximum average cumulative gain. However, the player does not know the probability distribution of machines, he estimates their average reward. For this, the player needs to explore the rewards of each machine to improve his estimation and select the one that offers him a better average gain. The more the phase of exploration is important, the more the regret of the player increases. Furthermore, a bad estimation and thus the choice of a bad machine to exploit leads to an increase of the regret.

The so-called Upper Confidence Bound (UCB) policies were shown to be asymptotically optimal. In particular, it is proven in Lai and Robbins [4] that such a policy incurs a regret of order $O(\log T)$, where T is the length of the time horizon. For that purpose, UCB policies estimate the expectation of each arm in a similar way to upper confidence bound, and the exploration expands further the sequences with highest UCB (See Algorithm 1).

Note that UCB policies are adapted for different types of application. For example, solving distributed constraint optimization problems using UCBs has been considered previously in Ottens et al. [5] (the so-called DUCT). In Kocsis and Szepesvari [6], where the resulting algorithm, UCT (UCB applied to Trees), has been successfully applied to the large scale tree search problem of computer-go, see Gelly et al. [7].

Algorithm 1 UCB

```
Require: $K$ the number of machines , $\tilde{x_j}$ the average reward
  of machine $j$, $T_j$ the number of time machine $j$ was selected
  Initialization: Explore each machine $j$ and set $\tilde{x_j}$
  while $t < T$ do
    for each machine $j$ do
      Set $index_j = \tilde{x_j} + \sqrt{\frac{2 \ln t}{T_j}}$
    end for
    Select machine $j$ that maximizes $index_j$
  end while
```

Theorem 3.1: [8] For all $K > 1$, if UCB is run on K machines having arbitrary reward distributions $P_1, \dots, P_K$ with support in $[0, 1]$, then its expected regret after any number T of plays is at most

$$\left(8 \sum_{i:\mu_i < \mu^*} \frac{\ln T}{\Delta_i} \right) + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j=1}^{K} \Delta_j \right) \tag{4}$$

where $\mu_1, \ldots, \mu_K$ are the expected values of $P_1, \ldots, P_K$ and $\Delta_i = \mu^* - \mu_i$

B. Adaptation to SLA failure detection model

In a MAB model of the problem, the customer is the player and he faces the NSPs representing the machines. The NSPs have a fixed number of offers with a fixed reward. However, when establishing an SLA between the customer and an NSP this one has a chance to fail. The failure modifying the reward received by the customer, each NSP has an associated unknown reward distribution. Moreover, when the customer chooses an NSP, the remaining capacity of this NSP decreases and on the contrary his failure rate grows. Then an NSP could have no more sufficient remaining capacity to propose an offersatisfying the customer. Hence, a notable difference with the assumptions MAB algorithms rely on lies in the fact that, in our problem, at each round, *one or more machines may not be available*. To take into account this peculiarity, we first focused on the UCB algorithm following the path of other authors [6], who used to consider first UCB as a candidate algorithm for modifying rather than more complex MAB algorithms.

We propose an adaptation of the definition of the regret to this context. At each round t, $j \in X(t)$ if machine j is available otherwise $j \notin X(t)$. The regret after T steps on K machines is :

$$R_T = \sum_{t=1}^{T} \max\{\mu_j : j \in X(t)\} - \sum_{j=1}^{K} \mu_j \times T_j \qquad (5)$$

The first proposed variation of the UCB algorithm is to simply select the first machine j such that $j = arg\max\{index_j : 1 \leq j \leq K\}$ even if j is not available. We called this algorithm the UCB-Adapted algorithm. Unfortunately, with this adaptation of the algorithm, the UCB theorem on expected regret is no longer appropriate.

In order to preserve the minimizing regret property we modify the algorithm as follows (see Algorithm 2 for a formal description). At round t, all machines in $X(t)$ are available. As an UCB algorithm, the index of a machine j is calculated from the estimated average reward of the machine, which is not time dependent, and an exploration factor. This exploration factor depends on the number of times the machine j was played (T_j) and the number of rounds when all machines in $X(t)$ were played. By using $\sum_{i \in X(t)} T_i$ instead of T for the actual number of rounds, the index calculated is the same as if the player was facing the $X(t)$ configuration of machines from the beginning to the time $T_{X(t)} = \sum_{i \in X(t)} T_i$. Therefore, for each subset X of $[1, K]$, we are able to bound the expected regret using the UCB theorem.

$$\left(8 \sum_{i:\mu_i < \mu^*} \frac{\ln T_X}{\Delta_i}\right) + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{j \in X}^{|X|} \Delta_j\right) \qquad (6)$$

where $\Delta_i = max\{\mu_j : j \in X\} - \mu_i$ for all $i \in X$.

Remark: To apply a MAB algorithm such as the UCB, the NSPs must have arbitrary reward distributions with a support in

Algorithm 2 UCB-Modified

```
Initialization: Explore each machine and observe the reward
while $t < T$ do
    for each machine $j$ such that $j \in X(t)$ do
        set $index_j = \tilde{x_j} + \sqrt{\frac{c \ln \sum_{i \in X(t)} T_i}{T_j}}$
    end for
    select $j$ s. t. $j = \arg\max\{index_j : 1 \leq j \leq K\}$ and
    observe the reward
end while
```

$[0, 1]$. Since the NSPs' offers received by the customer always fits the required QoS but may have different prices or QoS guarantees, the reward obtained by the customer needs to take both of them into account. We normalize the sum $\beta P_{c,k} + Q_{c,k}$ in order to obtain the following property $0 \leq rew_c(k, q_k) \leq 1$.

IV. NSP'S QOS OFFER ALGORITHM

We are interested in the behavior of the customer and in the maximization of his accumulated gain. Nevertheless the way an NSP selects an offer, to be proposed to the customer, among its whole set of offer affects this one. We tested the behavior of the customer when the NSPs use three different types of algorithms. These types of algorithms are a naive heuristic, a random selection and a learning algorithm.

We have, for NSP k, Q_k the set of QoS capabilities and Q'_k, such as $Q'_k \subseteq Q_k$, the set of QoS capabilities which satisfy the customer's requested QoS guarantees. When the customer selects the offer k with path i proposed by NSP k, this one gets the reward

$$rew_k = \frac{g_k(q_k, i) - g_i(q_k)}{maxGain_k}$$

where $maxGain_k$ is the higher price of QoS capabilities in Q_k. This reward function permits $0 \leq rew_k \leq 1$ which is needed for the learning algorithms described above. If the NSP is not selected by the customer or if there is a fail when the SLA is established then its reward is $rew_k = 0$

Naive heuristic : the "Discounter": The NSP computes the price of all QoS capabilities in Q'_k. At each round he provides the customer with the offer of cheaper price.

Random selection: The NSP selects an offer uniformly at random in the set of QoS capabilities Q_k. This set of QoS capabilities contains all the QoS offers, including the ones who do not satisfy the customer's requested QoS.

Learning algorithms: The objective is to learn a selection policy according to the previous noticed failures or successes. We opt for two different learning algorithms : Linear Reward Inaction (LRI) [9] and Upper Confidence Bound (UCB).

For UCB, the algorithm is used with no modification. The behavior is the same as described in Section III.

Here the player is the NSP and the gambling machines are the offers of the set of QoS capabilities Q_k. When choosing a machine corresponding to offer k with path i the NSP gets the reward rew_k with $0 \leq rew_k \leq 1$. Since the reward depends on the offer and the failure each machine has an unknown reward distribution.

The LRI algorithm uses a probability vector for the selection of the next action. Each action is associated with a choice probability. Initially, this probability is the same for all the actions. According to the reward obtained at each round, the probability vector is updated depending on a factor $b \in [0, 1]$. Depending on the value of the b factor, the algorithm converges more or less slowly to a solution. This factor permits an increase of the probability of selected actions where higher rewards have been observed.

The actions are the offers of the set of QoS capabilities Q_k of NSP k. Let $P_k(t)$ be the probability vector at time t and thus $P_{i,k}(t)$ the probability of action i to be selected.
For any offer $i \in Q_k$:

$$P_{i,k}(t+1) = P_{i,k}(t) \begin{cases} +b.rew_k(t)(P_{i,k}(t)) \text{ if } i \text{ selected} \\ -b.rew_k(t)(P_{i,k}(t)) \text{ otherwise} \end{cases} \tag{7}$$

The algorithm used for each NSP k is described by Algorithm 3.

Algorithm 3 Linear Reward Inaction Algorithm

```
Initialization: for all offers $i \in Q_k$, set $P_{i,k}(0) = \frac{1}{|Q_k|}$
while $t < T$ do
    Select offer according to $P_{i,k}(t)$ and observe the reward
    for each offer $i \in Q_k$ do
        update the probability vector using eq. (7)
    end for
end while
```

V. Experiments

The simulations were performed on the network example presented in Figure 1. At each decision epoch, customer c sends a request $q_c = (250\text{Mbps}, 20\text{ms}, 0.5\%)$ with duration $\Delta_c = 10$. The reward function used by the customer is the one described by eq. (2) with $\beta = 0.5$. With these parameters, the price bound computed for the required QoS is $g_c(q_c) = 2439.02$. Each QoS offer with a price higher than 2439.02 is rejected.

The NSPs (NSP1, NSP2 and NSP3 in Figure 1) have a fixed capacity of 3000 Mbps. Each NSP has two disjointed paths, one with a price bound of 1 (type 1) and the other of 5 (type 2). Those two paths have the same capacity as the NSPs and both have the same failure rate : 0.1 for NSP1, 0.3 for NSP2 and 0.6 for NSP3. As explained at the end of Section II, the NSP has two different types of failure and this is the failure rate for the NPS's external path to the destination which have a fixed one. In the following we consider experiments with failure on paths only (with a fixed probability of failure on each path) and with both capacity and path failure. Table I details the offers of each NSP.

We added 4 QoS capabilities not appearing in Table I with a price higher than the customer maximum price for NSP2 and NSP3. With these offers the maximum reward the customer can obtain by choosing NSP1 is 0.407, 0.593 for NSP2 and 0.697 for NSP3.

TABLE I
NSPs offers

				Path type 1		Path type 2	
	b_k	d_k	l_k	$g_k(q_k)$	rew_c	$g_k(q_k)$	rew_c
NSP1	250	20	0.5	134.14	0.315	182.92	0.308
	500	20	0.5	268.29	0.407	365.85	0.394
	2000	20	0.5	1073.17	0.381	1463.41	0.327
	3000	20	0.5	1609.75	0.317	2195.12	0.237
NSP2	750	10	0.25	804.87	0.593	1097.56	0.553
	1000	10	0.25	1073.17	0.575	1463.41	0.522
	2000	20	0.1	1094.52	0.555	1492.53	0.501
	3000	20	0.01	1649.17	0.529	2248.87	0.447
NSP3	250	5	0.01	548.90	0.642	748.50	0.615
	500	20	0.01	274.86	0.624	374.81	0.611
	750	10	0.01	824.17	0.697	1123.87	0.656

To compare the results of the algorithms, we used the three variants of UCB described in Section III (UCB-Basic, UCB-Adapted and UCB-Modified), the Best Response algorithm and a utility based algorithm (this latter selects the offer having the highest utility or randomly in case of equality). Formula (2) is used as a utility except that this one is multiplied by the NSP reputation factor defined by Lamali et al [3]. When using the reputation algorithm, the customer takes into account his own experience about past SLA failure in order to determine the NSPs' reputation. The reputation factor is computed at each time t and is defined as :

$$rep_k(t) = 1 - \frac{\#fail_k(t)}{\#choice_k(t)}$$

for NSP k, where $\#fail_k(t)$ is the number of times the customer has selected an offer of NSPk which has then been violated, and $\#choice_k(t)$ is the number of times the offers of NSPk were selected by the customer.

The UCB algorithm is also compared with the LRI described in Section IV. The same changes to UCB were applied to the LRI algorithm in order to capture the availability of NSPs. There are three variants of the LRI algorithm, the LRI-Basic (described in Section IV), the LRI-Adapted (it selects an offer according to the vector of probabilities, until finding an available NSP) and the LRI-Modified (equivalent to the LRI-Adapted with a normalized vector according to the NSP availability).

All results were obtained for 100 simulations of $20,000$ requests. Table II summarizes the various scenarios and the used NSPs' selection algorithms.

TABLE II
NSPs selection policy

	NSP 1	NSP 2	NSP 3
Scenario 1	Random selection		
Scenario 2	The Discounter		
Scenario 3	Learning algorithm : LRI		
Scenario 4	Learning algorithm : UCB		

Comparison of the UCB algorithms: Figure 2 represents the customer's mean cumulated reward over 100 simulations. The simulations correspond to scenario 1 of Table II and are run for two types of SLA failure. Simulations ”Path failure” on Figure 2 use, for SLAs failure, only the path probability of failure whereas simulations ”Capacity failure” use both the capacity function and the path probability of failure. Regardless

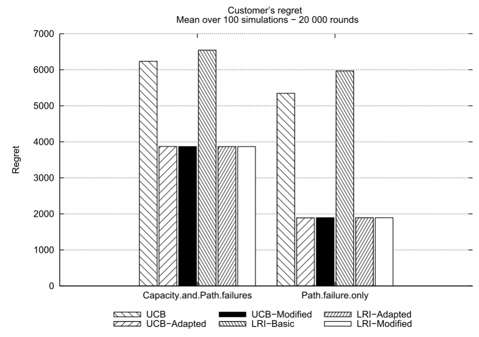


Fig. 2. Comparison of the UCB algorithms in scenario 1 - Customer's regret

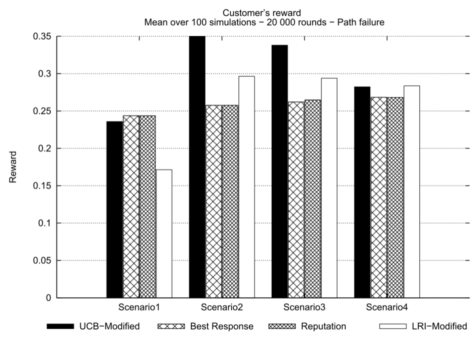
of studied case, the customer's regret is better for the variations of UCB than for the seminal algorithm. This is due to the fact that the seminal algorithm can select “dummy” offers (for instance when an NSP has no more capacity and does not make an offer). For both versions of UCB, UCB-Adapted and UCB-Modified, the regret as well as the reward is similar. Nevertheless, recall from eq. (5) that the UCB-Adapted version does not allow predicting this behavior.

Learning algorithms modified and the utility based algorithms: Given the results of Figure 2, we realized further simulations only on the UCB-Modified and LRI-Modified algorithms. In Figures 3(a) and 3(b) the histograms are the mean cumulated regret of the customer.

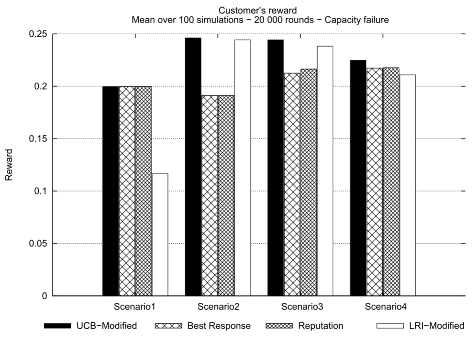
Whether it is with or without the failure according to the remaining capacity of the NSP, for the four scenarios described in Table II the UCB-Modified algorithm gives better results. Simulation results on scenario 2 exhibit a better efficiency of the UCB-Modified algorithm. Indeed the NSPs use the algorithm ”Discounter” and therefore each NSP always proposes the same offer, the lowest price one. Therefore a customer's decision algorithm based on the utility always chooses the same NSP and increases its failure rate, inducing a bad profit.

We can thus look at the capacity of the UCB-Modified algorithm to detect the failure rate of the NSPs. For this we observe the percentage of requests where the customer has established an SLA which has suffered a failure. Figures 4(a) and 4(b) show the percentage of failed requests. We notice that for each type of SLA's failure and scenario of Table II, the algorithm UCB-Modified obtains a percentage of failure lower than the two others.

The algorithm UCB-Modified is able to learn which NSP offers the best reward with less probability to fail when establishing an SLA. The customer's ability to learn the reliability of NSPs could also be observed in Tables III and IV. Table III contains the results of simulations with capacity and path violation whereas Table IV contains the ones with path violation only.



(a) With failure on paths only



(b) With failure both on capacity and paths

Fig. 3. Customer's reward

TABLE III
CUSTOMER'S CHOICES AND NSPS' FAILS - SIMULATIONS WITH CAPACITY AND PATH VIOLATION

Scenario 2

	UCB-Mod.		Best resp.		Rep.		LRI-Mod.	
%	Select	Fail	Select	Fail	Select	Fail	Select	Fail
NSP1	58	18	0	-	0	-	88	20
NSP2	30	59	1	29	1	33	10	34
NSP3	12	67	99	69	99	69	2	60

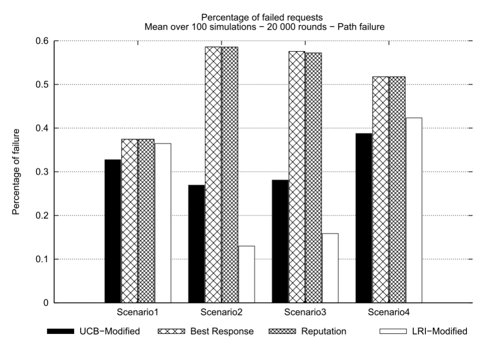
Scenario 3

	UCB-Mod.		Best resp.		Rep.		LRI-Mod.	
%	Select	Fail	Select	Fail	Select	Fail	Select	Fail
NSP1	56	19	1	17	1	17	96	24
NSP2	22	60	1	40	1	47	3	64
NSP3	22	63	98	67	98	66	1	60

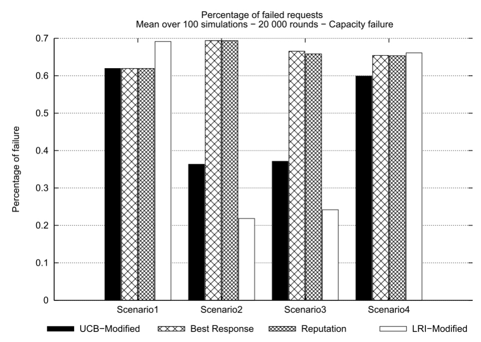
These tables list the percentage of times each NSP has been selected by the customer and the associated percentage of failures when the customer uses the algorithms UCB-Modified, Best Response, Reputation and LRI-Modified. For Table IV the failure rate is not mentioned. Indeed, the violation is on paths only and paths have a fixed failure rate : 0.1 for NSP1, 0.3 for NSP2 and 0.6 for NSP3.

The results are given for both scenarios 2 and 3 due to the non-random behavior of the NSPs in those. Indeed, in scenario 2 the NSPs use the ”Discounter” algorithm which implies that they always propose the same offer, the one with the cheapest price. For scenario 3 the NSPs selection algorithm is LRI-Modified and permit to converge to propose the best offer.

Remember that, according to Table I, the maximum reward obtained by the customer when choosing NSP1 is 0.407, 0.593



(a) With failure on paths only



(b) With failure on both capacity and paths

Fig. 4. Percentage of failed requests

TABLE IV
CUSTOMER'S CHOICES AND NSPS' FAILS - SIMULATIONS WITH PATH VIOLATION ONLY

Scenario 2

	UCB-Mod.	Best resp.	Rep.	LRI-Mod.
%	Select	Select	Select	Select
NSP1	34	0	0	88
NSP2	53	5	5	10
NSP3	12	95	95	2

Scenario 3

	UCB-Mod.	Best resp.	Rep.	LRI-Mod.
%	Select	Select	Select	Select
NSP1	36	1	1	78
NSP2	46	7	7	12
NSP3	18	92	92	10

for NSP2 and 0.697 for NSP3. Whatever the type of SLA violation, we notice that for both scenarios, when the customer uses utility based algorithms, this one selects the NSP with the maximum reward, which is NSP3, but its failure rate is the most important. In contrast, when the customer uses learning algorithms such as UCB-Modified and LRI-Modified, we can notice that a compromise between the reward and the failure rate is obtained. Indeed we observe in Table III that the customer selects the NSP1 which has the minimum failure rate among NSPs. In Table IV we remark a different behavior for LRI-Modified and UCB-Modified algorithms. The LRI-Modified algorithm converges to the selection of NSP1 which has the lowest failure rate, while the UCB algorithm is shared between NSP1 and NSP2. This is due to the fact that the LRI algorithm updates gradually a probability vector, for the selection of NSP, based on previous reward until one component reaches 1. On the contrary the UCB-Modified is not permanently fixed on the choice of one NSP but uses all the information about previous rewards to make its choice. It obtains a compromise between the NSP2 which has a slightly higher failure rate than the NSP1, but balanced by a more attractive reward.

These results reinforce the belief that learning algorithms enable the customer to estimate the reliability of NSPs.

VI. CONCLUSION

In this paper, we provided a model for the problem faced by a customer when choosing a QoS-offer that can encounter failures in a context where providers might change (due to capacity or other constraints, they do not make offers). This problem is equivalent to the one of a player having a variable utility facing dynamic machines in a Multi-Armed Bandit problem. We proposed an adaptation of the UCB and LRI algorithms to address this problem.

Surprisingly, the simulation results exhibit that the NSPs behave as reward distribution machines. Our proposed algorithms are adapted to such configuration (for instance it detects discounting behaviors). In future work, we will consider adapting some other well-known Multi-Armed Bandit algorithms to SLA negotiation.

REFERENCES

[1] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.
[2] R. Douville, J. L. Roux, J. Rougier, and S. Secci, "A service plane over the PCE architecture for automatic multi-domain connection-oriented services," *IEEE Communication Magazine*, 2008.
[3] M. L. Lamali, D. Barth, and J. Cohen, "Reputation-aware learning for sla negotiation," in *Networking Workshops*, 2012, pp. 80–88.
[4] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules." *Advances in Applied Mathematics*, pp. 4–22, 1985.
[5] B. Ottens, C. Dimitrakakis, and B. Faltings, "Duct: An UCB approach to distributed constraint optimization problems," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
[6] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *17th European Conference on Machine Learning, Berlin, Germany, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4212, 2006, pp. 282–293.
[7] S. Gelly, Y. Wang, R. Munos, and O. Teytaud, "Modification of UCT with Patterns in Monte-Carlo Go," INRIA, Rapport de recherche RR-6062, 2006.
[8] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, 2002.
[9] P. S. Sastry, V. V. Phansalkar, and M. A. L. Thathachar, "Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information," *IEEE Trans Systems, Man, and Cybernetics*, vol. 24, 1994.