Risk-Aware SLA Negotiation*

Mohamed Lamine Lamali
Alcatel-Lucent Bell Labs France
Route de Villejust
91620, Nozay. France
mohamed_lamine.lamali@alcatel-lucent.com

Hélia Pouyllau†
Thales Research & Technology France
1 av. A. Fresnel. 91767
Palaiseau Cedex
helia.pouyllau@thalesgroup.com

Johanne Cohen
Lab. PRiSM, UMR8144,
Université de Versailles
45, av. des Etas-Unis
78035 Versailles Cedex, France
johanne.cohen@prism.uvsq.fr

Anne Bouillard‡
ENS / INRIA
45, rue d'Ulm
75230 Paris Cedex 05, France
anne.bouillard@ens.fr

Dominique Barth
Lab. PRiSM, UMR8144,
Université de Versailles
45, av. des Etas-Unis
78035 Versailles Cedex, France
dominique.barth@prism.uvsq.fr

ABSTRACT

In order to assure Quality of Service (QoS) connectivity, Network Service Providers (NSPs) negotiate Service Level Agreements (SLAs). However, a committed SLA might fail to respect its QoS promises. In such a case, the customer is refunded. To maximize their revenues, the NSPs must deal with risks of SLA violations, which are correlated to their network capacities. Due to the complexity of the problem, we first study a system with one NSP provider and give a method to compute its risk-aware optimal strategy using $(\max, +)$-algebras. Using the same method, we study the case where two NSPs collaborate and the case where they compete, and we derive the Price of Anarchy. This method provides optimal negotiation strategies but, when modeling customers' reaction to SLA failure, analytical results do not hold. Hence, we propose a learning framework that chooses the NSP risk-aware optimal strategy under failures capturing the impact of reputation. Finally, by simulation, we observe how the NSP can benefit from such a framework.

*This work has been partially supported by the ETICS-project, a project funded by the European Commission through the 7th ICT-Framework Program. Grant agreement no.: FP7-248567 Contract Number: INFSO-ICT-248567.

†This work was realized when Hélia Pouyllau was affiliated with Alcatel-Lucent Bell Labs France.

‡Anne Bouillard has carried out the work presented in this paper at LINCS (www.lincs.fr).

Keywords

SLA negotiation, Quality of Service, Reputation, $(\max, +)$-algebras, Markov chains, Learning Algorithms.

1. INTRODUCTION

Internet actors, and in particular Network Service Providers (NSPs), are facing an increasingly network resource demand from Content Service Providers (CSPs) and end-users. Current Internet applications (e.g. video streaming, gaming, telepresence, etc.) impose more and more constraints on the networks, such as high bandwidth, low delay and packet-loss, etc. Guarantees on such parameters are defined as Quality of Service (QoS) guarantees. Service Level Agreements (SLAs) comprise such guarantees (bandwidth, delay, etc.) and charging conditions (price, penalties, etc.). They are bilaterally negotiated between a *customer* (which can be a company, a CSP or another NSP) and a *provider* (an NSP).

Upon the reception of a customer's request, an NSP faces then several issues: first, it must identify whether its network resources can sustain or not the required QoS. And second, it has to provide a price that will allow it to be selected by the customer. Solving these issues is particularly challenging since: *i*) the NSP has no information on the proposal of its competitors, *ii*) the customer might demand a QoS that places too many constraints on its resources, forbidding further resources with higher willingness to pay, *iii*) the NSP has no information on the customer behavior: his willingness-to-pay, his QoS sensitivity (some customer might prefer to be upgraded if the cost difference is not so high), his possible preferences among providers (i.e customer's sensitivity to the NSP *reputation*) and his request arrival law. Furthermore, the NSP proposal (and how its commitment on the SLA is achieved if it is chosen) will impact both its network resources and the customer behavior.

A committed and provisioned SLA can fail, i.e., its QoS guarantees are violated. Such a violation leads to a loss of

revenues and negatively impacts the reputation of the NSP. Each NSP having a limited *capacity*, the more its capacity is used, the higher is the failure probability. Thus, when provisioning the resources for an SLA, the NSP should be aware of the impact of the provisioned capacity on the failure probability. As the SLAs are provisioned for a given **time period**, the service duration affects the network capacity. The goal of an NSP is thus to find the *optimal trade-off* (also called negotiation strategy) between selling too few SLAs (and thus keeping a good reputation but gathering little revenues) and selling too many SLAs (and thus gathering more immediate rewards but decreasing its reputation – which leads to decreasing its long term revenues). Having a system that computes optimal negotiation strategies can thus bring a highly competitive advantage to NSPs. Considering the analysis of the SLA negotiation, several works ([1, 2, 3, 4]) focused on the same parameters influencing the customer's choice: sensitivity to the QoS parameters and price, proposals from competitor NSPs. Very few works considered the influence of the failure and the reputation on the customer's choice. In a previous work [5], we introduced a simple model of reputation and argued how the NSPs should take into account the customer's reputation sensitivity.

In [5], the introduced model did not capture the *impact of the time-period* during which an SLA is provisioned. Such time constraints add a complexity dimension to the risk-aware SLA negotiation problem. In this paper, to illustrate the complexity of the problem, we first study a system of one customer and one NSP. We give a method to compute the optimal strategy of the NSP using $(\max, +)$-algebra tools. We then study a system of one customer and two NSPs without reputation (as illustrated by Fig. 1) and provide a method to compute the optimal strategy when the NSPs collaborate and when they are selfish, and we discuss the Price of Anarchy in such a system. When introducing the reputation, the system becomes too complex to be analytically studied. Furthermore, the failure probability and its impact on the reputation of the NSP are not known in practice. Thus, we opt for a framework of learning algorithms applied to the SLA negotiation problem with reputation and time-period reservation and perform simulations to assess the ability of these algorithms to learn risk-aware strategies.

The paper is organized as follows: Section 2 presents the SLA negotiation problem and summarizes some related works. Section 3 describes the model of the system. Section 4 describes a method to compute the optimal strategy of one NSP when facing the requests of one customer. Section 5 studies a system comprising one customer and two NSPs (if they collaborate and if they are selfish). Section 6 introduces a reputation mechanism and describes how it impacts the NSP optimal strategy. Section 7 describes the learning framework extended to support reputation and service duration. Finally, section 8 presents the simulation results obtained with the learning algorithms.

2. LONG TERM PROVISIONING AND RELATED WORK

In the studied scenario, depicted by Fig. 1, the NSPs compete for the customer's selection through SLA offers.

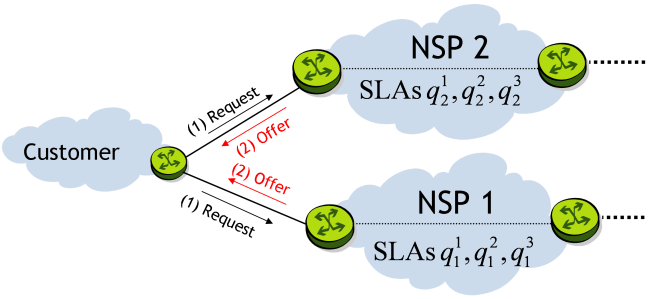

Figure 1: Competition between two NSPs to be selected by a customer

2.1 SLA negotiation

As in several frameworks [6, 2], the customer first sends a request specifying required QoS parameters (e.g. bandwidth $>$ 10Gbps and delay $<$ 10ms) to its neighbor NSPs. Each NSP then sends an offer to the customer. This offer can satisfy or exceed the QoS parameters requited by the customer. Each NSP also sends a price corresponding to its offer. Finally the customer selects one offer which will be instantiated by the provider NSP. The NSPs compete on QoS parameters and prices in order to be selected by the customer. We assume in this paper that the SLAs have the same duration, and that each NSP receives a single request at each time. However, since the SLA duration exceeds one time unit, the provisioned SLAs of the same NSP can overlap. Fig. 1 illustrates two NSPs (each one having 3 SLAs) in competition to be selected by a customer.

The NSP is facing two important challenges: i) which SLA the NSP would offer in order to be selected by the customer, ii) how to minimize the failure probability and keep a good reputation. Achieving these challenges is difficult for several reasons: the NSP has no information on the customer's profile (customer's sensitivity to the QoS, to the price, to the reputation, etc.). The NSP has no information on the offers of the other NSPs, neither their SLA violation probability. It is not able to anticipate the future requests and its future network state (future used capacity, reputation, etc.). The purpose of our approach is to provide an algorithm that, avoiding the use of forecasting method, will embed observation from both customer behavior and network state to recommend an SLA offer to propose.

2.2 Related work

The SLA negotiation problem and risk-aware provisioning are generally separately considered. In [2], the authors propose a Reinforcement Learning framework to perform SLA negotiation and insure end-to-end QoS guarantees. The authors of [7] analyze the SLA negotiation problem in a game theoretic perspective. They specify the strategies that allow to reach an equilibrium. In a previous work [3], we presented a framework of Reinforcement Learning algorithms applied to the SLA negotiation problem and compared their performances in simulation results.

Considering the risk-aware provisioning problem, the authors of [8] propose a provisioning scheme which minimizes the SLA violation risk and allows a minimum risk path selec-

tion in WDM[1] networks. The authors argue that taking into account the statistical availability of a path is not sufficient to minimize the violation risk. They propose to take into account more accurate parameters (such as failure profile, availability target, etc.) in the path computation scheme. Other works [9, 10] focus on management risk mechanisms to assure QoS guarantees in Grid Systems.

These works do not make a relation between the SLA negotiation (competing to be selected by the customer) and long term and risk-aware provisioning (which provisioning strategy adopt in order to keep a low failure probability). A way to connect these two problems is to introduce a reputation mechanism (over the NSPs) which is dependent upon the NSP failure rate (i.e., SLA violation rate) and to consider that the customer is sensitive to the reputation. In a previous work [5], we provide such a mechanism and present simulation results showing that Reinforcement Learning algorithms can learn strategies that keep good reputation. However, the temporal aspects of the SLA negotiation (SLA duration, SLA overlapping and its impact on the future available resources, etc.) were not considered. These aspects complicate the problem of long-term provisioning.

In this work, we first study the case where the SLAs have the same (fixed) duration and arrive at regular intervals. We provide a method to compute the NSP optimal strategy when there is one NSP, and then when there are two NSPs. In the latter case, we also compute the optimal strategy when the NSPs are selfish and derive the Price of Anarchy. In the second part of this work, we introduce a reputation mechanism and a learning framework to perform SLA negotiations with reputation.

3. RISK-AWARE SLA NEGOTIATION MODEL

3.1 NSP characteristics

Each NSP has a fixed maximum capacity denoted by c_{max}. Let c_t be the NSP capacity used between decision epoch $t-1$ and decision epoch t. We define the used capacity rate at decision epoch t as $\rho_t = c_t/c_{\max}$ and the average used capacity rate (during some period T) as $\tilde{\rho} = (\sum_{t=1}^{T} \rho_t)/T$.

An SLA proposed by NSP i is denoted by q_i^j. It is a 3-tuple (b_i^j, d_i^j, ℓ_i^j) where b_i^j is the bandwidth, d_i^j the delay and ℓ_i^j the packet-loss. All the SLAs have the same duration Δ. The set of SLAs of NSP i is denoted by $\mathcal{Q}_i$.

3.1.1 SLA unit price

Each NSP set a unit price p_u. Thus, the price of an SLA q_i^j per unit of time is $p_u \cdot b_i^j$ and the total price of an SLA is $p_u \cdot b_i^j \cdot \Delta$.

3.1.2 The Failure function

The probability of failure of an SLA depends on the used capacity rate of the NSP. As a function of the used capacity rate, it is denoted by $f(\rho_t)$. The failure probability can be for example $f(\rho_t) = \rho_t$, $f(\rho_t) = \rho_t^2$ or $f(\rho_t) = 1 - \exp(-\rho_t)$.

[1]Wavelength Division Multiplexing.

3.1.3 NSP reward

An NSP get a reward at each decision epoch. The reward obtained at decision epoch t is denoted by v_t. If c_t is NSP used capacity at decision epoch t then:

$$v_t = \begin{cases} p_u \cdot c_t = p_u \cdot c_{\max} \cdot \rho_t & \text{if there is no failure} \\ & \text{at decision epoch } t \\ 0 & \text{otherwise} \end{cases}$$

Thus, the expected reward at decision epoch t is $\mathbb{E}(v_t) = p_u \cdot c_{\max} \cdot \rho_t \cdot (1 - f(\rho_t))$.

3.2 Customer's characteristics

In order to compare the customer's sensitivity to the QoS parameters and to the price, the QoS parameters are regrouped within a normalized measure in $[0, 1)$.

3.2.1 Quality of Service measure

A measure $||\cdot||$ on the QoS provided by an SLA is computed according to its parameters. It is defined as:

$$||\mathrm{q}_i^j|| = 1 - \frac{1}{3}\left(\frac{\ell_i^j}{\ell_{\mathrm{req}}} + \frac{d_i^j}{d_{\mathrm{req}}} + \frac{b_{\mathrm{req}}}{b_i^j}\right),$$

where b_{req} is the minimum bandwidth requested by the customer, d_{req} the maximum delay and ℓ_{req} the maximum packet-loss. When q_i^j complies with the QoS requirements, then $||\mathrm{q}_i|| \in [0, 1)$. The function $||.||$ tends to 1 when the values of QoS parameters of the SLA surpass the customer's requirements. The SLAs that do not comply with the customer's requirements are not considered.

3.2.2 Customer's utility according to an SLA

We opted for the classical customer's utility (Quality of Service − price):

$$U(\mathrm{q}_i^j) = ||\mathrm{q}_i^j|| - \eta \frac{p_i^j}{p_{\max}}$$

where η is a fixed parameter, p_i^j is the price of q_i^j, and $p_{\max}$ is the maximum price the customer is willing to pay.

4. ONE NSP WITHOUT COMPETITION

Here we study the problem where there is only one NSP (called NSP 1) and one customer. The SLAs are simply denoted by q^j and their parameters by b^j, d^j and ℓ^j.

4.1 The NSP states

In order to select the SLA that maximizes its average reward, NSP 1 needs a complete information on its current and future available capacity. At each decision epoch t, NSP 1 knows which SLAs it provisioned until decision epoch t. While the SLAs provisioned before decision epoch $t - \Delta$ are already released, those provisioned between decision epochs $t - \Delta$ and t are still using the NSP capacity. Thus, these SLAs allow to know how much capacity will be released at each decision epoch between t and $t + \Delta$.

Thus, we define the *state* of NSP1 at decision epoch t as the ordered list of SLAs That will be released between t and $t + \Delta$. In fact, the NSP only needs to know their bandwidth. Thus, we define more formally the state of NSP1 at decision epoch t as a vector $s_t = (b^1, \ldots, b^\Delta)$ where b^j is the bandwidth of the SLA that will be released at decision

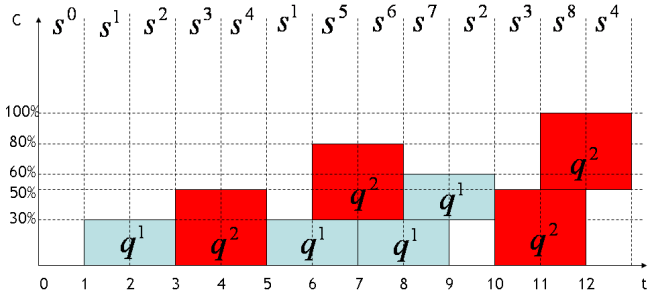

Figure 2: A digram showing the different states of NSP 1.

epoch $t+j$. Thus, the used capacity at decision epoch t is $c_t = \sum_{i=1}^{\Delta} b^j$.

It is clear that if NSP 1 total capacity $c_{\max}$ is unbounded then the total number of possible states of NSP 1 is $(|\mathcal{Q}|+1)^{\Delta}$. However, since $\forall t,\ c_t \leq c_{\max}$, the states $s' = (b'^1, \ldots, b'^{\Delta})$ for which $\sum_{j=1}^{\Delta} b^j > c_{\max}$ are not valid (we mean by *not valid* that these states require to use a capacity greater than the total capacity of NSP 1. Thus, they cannot be reached). Let $\mathcal{S}_1$ be the set of valid states of NSP 1. Then:

$$(|\mathcal{Q}|+1)^{\min(\Delta,K)} \leq |\mathcal{S}_1| \leq (|\mathcal{Q}|+1)^{\min(\Delta,K')},$$

where $K = \lfloor c_{\max} / \max_{\mathrm{q}^j \in \mathcal{Q}_1} b^j \rfloor$ and $K' = \lfloor c_{\max} / \min_{\mathrm{q}^j \in \mathcal{Q}_1} b^j \rfloor$.

Example 1.1 This example shows how the states of an NSP are defined with respect to its SLAs. Let the maximum capacity of NSP 1 be $c_{\max} = 100$Gbps. It has two SLAs: q^1 with bandwidth $b^1 = 30$Gbps$= 30\%.c_{\max}$ and q^2 with bandwidth $b^2 = 50$Gbps$= 50\%.c_{\max}$. Thus, its set of strategies is $\{\mathrm{q}^1, \mathrm{q}^2\}$. The duration of both SLAs is $\Delta = 2$ time units. The diagram on Fig. 2 shows the different possible states of NSP 1. Each possible combination and overlapping of the SLAs define a state. For instance, $s^5 = (b^1, b^2)$ is a state where q^1 will be released at $t+1$ and q^2 will be released at $t+2$. In this example, $|\mathcal{S}_1| = (|\mathcal{Q}_1|+1)^{\Delta} = (2+1)^2 = 9$. All the states are reachable.

Remark 2. Example 1.1, as the other examples in this paper, are obviously not realistic. Their purpose is to illustrates the approaches and the methods used in this paper rather than to provide realistic data. In a concrete case, the SLAs are more fine-grained (mainly less than 5% of the NSP capacity) and have a longer duration. Such examples are not convenient to explain the methods described here because the number of states may be huge and can not be described clearly and concisely in a paper format. In Section 8, we use more realistic data to perform simulations.

4.2 Maximizing the average reward

At each decision epoch t, NSP 1 receives a request from the customer, then selects an SLA and offers it to the customer. NSP 1 may also offer nothing. The customer accepts the offer (because it is the only one). In this case the offered SLA is provisioned. If NSP 1 does not make an offer, it only releases the SLA provisioned at $t-\Delta$ (if it exists).

As a consequence, the transition from the state s_t to the state s_{t+1} is done by releasing the SLA provisioned at $t-\Delta$ (if it exists) and provisioning the SLA offered at time t (if it exists). The expected reward for NSP 1 is then $v_{t+1} \cdot (1 - f(\rho_{t+1}))$.

Define the transition function $\mathcal{T}_1 : \mathcal{S}_1 \times \mathcal{S}_1 \to \overline{\mathbb{R}}$ as $\mathcal{T}_1(s^i, s^j) = v_j \cdot (1 - f(\rho_j))$ for any possible transtion from s^i to s^j and $\mathcal{T}_1(s^i, s^j) = -\infty$ if the transition from s^i to s^j is not possible.

Let $X_t(s^i)$ be the optimal reward after t steps starting from state s^i. It can be computed by induction, using the linearity of the expectation, as: $X_0(s^i) = 0$ for all $s^i \in \mathcal{S}_1$ and

$$X_{t+1}(s^i) = \max_{s^j \in \mathcal{S}_1} \mathcal{T}_1(s^i, s^j) + X_t(s^j). \tag{1}$$

Equation (1) is linear in term of $(\max, +)$-algebras: see $\mathcal{T}_1$ as a square matrix and X_t as a column vector of size $|\mathcal{S}_1|$. Then we can write:

$$X_{t+1} = \mathcal{T}_1 \otimes X_t$$

and more generally,

$$X_{t+1} = \mathcal{T}_1^{t+1} \otimes X_0,$$

where for any matrices A and B, any vector X,

$$\begin{aligned} A \otimes X(s^i) &= \max_{s^k \in \mathcal{S}_1} A(s^i, s^k) + X(s^k) \\ A \otimes B(s^i, s^j) &= \max_{s^k \in \mathcal{S}_1} A(s^i, s^k) + B(s^k, s^j) \\ A^1 = A \quad &\text{and} \quad A^{n+1} = A \otimes A^n. \end{aligned}$$

More details about $(\max, +)$-algebras can be found in [11].

Now, the mean optimal reward is:

$$\mathbb{E}(v_{\max}) = \lim_{t \to \infty} \frac{(\mathcal{T}_1^t \otimes X_0)(s^0)}{t}.$$

This value exists and can easily be computed using the spectral theorem of $(\max, +)$ matrices [11]. In our setting, $\mathcal{T}_1$ is an irreducible (its graph is strongly connected) and aperiodic ($\mathcal{T}_1(s^0, s^0) \neq -\infty$) matrix, so there exist $\lambda \in \mathbb{R}$ and $T \in \mathbb{R}_+$ such that for all $t \geq T$, $\mathcal{T}_1^{t+1} = \lambda + \mathcal{T}_1^t$. Then λ is the desired value $\mathbb{E}(v_{\max})$, which can be computed as the average weight of a cycle of maximum average weight:

$$\mathbb{E}(v_{\max}) = \max_{k=1..|\mathcal{S}_1|} \frac{\max_{s \in \mathcal{S}_1} (\mathcal{T}_1^k)(s, s)}{k} \tag{2}$$

Formula (2) suggests a naive algorithm to compute $\mathbb{E}(v_{\max})$ in time $O(|\mathcal{S}_1|^4)$. However, using Karp's algorithm [12] reduces the time complexity to $O(|\mathcal{S}_1|^3)$ in the worst case.

As a consequence, NSP 1 will follow one of the cycle with maximum average weight in order to maximize its rewards.

Example 1.2 This example shows the graph of states and transitions of an NSP and the maximum average weight cycle which maximizes the NSP rewards. Let NSP 1 has the same SLAs as in Example 1.1. Thus, its set of strategies is $\{\mathrm{q}^1, \mathrm{q}^2\}$. Let the failure function be $f(\rho) = \rho$, $\rho \in [0, 1]$ (the failure probability is linear w.r.t. the used capacity rate). And let the unit price be $p_u = 1$ monetary unit (m. u.)

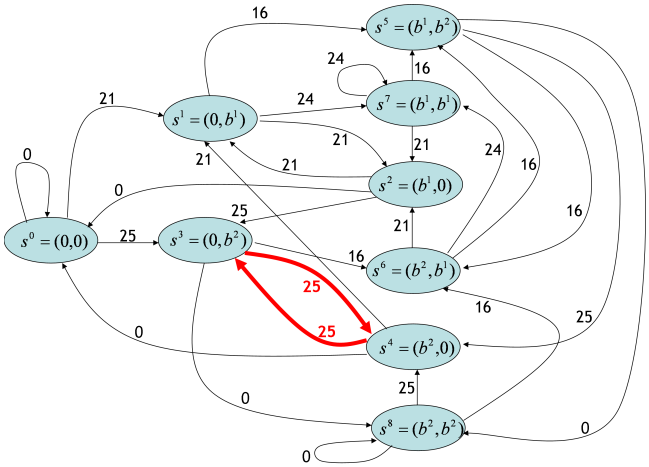

Figure 3: The graph of states and transitions associated to NSP 1. Each b^j is an SLA bandwidth. The cycle in bold is the cycle of maximum average weight.

per Gbps. Fig. 3 shows the set of states and the transitions associated to the NSP. The transitions are valuated by the expected reward when reaching the new state. The maximum average cycle is given by $s^3, s^4, s^3 \ldots$ The maximum average reward is 25 and the optimal used capacity rate is $\rho_{\text{opt}} = 50\%$.

4.3 Another approach to compute the maximum average reward

Suppose that NSP 1 can set its average used capacity rate $\tilde{\rho} = \tilde{c}/c_{\max}$ (where $\tilde{c}$ is the average used capacity) to any value in $[0, 1]$. And suppose that it knows the value of the failure function $f(\tilde{\rho})$ for all $\tilde{\rho} \in [0, 1]$. The NSP reward, as defined in Section 3.1.3, depends on the bandwidth of the provisioned SLAs.

$$v = p_u \cdot c_{\max} \cdot \tilde{\rho}$$

Thus, the average reward per time unit is given by $v(1-f(\tilde{\rho}))$ and NSP 1 goal is to compute the optimal average reward, which is $\rho_{\text{opt}} = \arg\max_{\tilde{\rho}\in[0,1]} v(1 - f(\tilde{\rho}))$.

If NSP 1 knows the values of $f(\tilde{\rho})$ and v for all $\tilde{\rho} \in [0, 1]$, then it is easy to compute the optimal average used capacity that maximizes the average reward. This approach is useful if the SLAs are fine-grained, which allows to closely approximate ρ_{opt} by $(\sum_{t=1}^{\Delta} c_t)/\Delta$

Example 1.3. Let, as in the previous examples, the maximum capacity of NSP 1 be $c_{\max} = 100$Gbps, the unit price be $p_u = 1$ m.u. and the failure function $f(\tilde{\rho}) = \tilde{\rho}$. The maximum average reward $\mathbb{E}(v_{\max})$ is obtained by maximizing the function $v(1 - f(\rho))$. The maximum value is obtained when $\rho = 0.5$. The maximum average reward is $\mathbb{E}(v_{\max}) = 25$ m.u. and the optimal average used capacity is $\rho_{\text{opt}} = 50\%$. It exactly corresponds to the maximum weight of a cycle obtained in Example 1.2.

5. TWO NSPS AND ONE CUSTOMER

In this section we focus on the SLA negotiation problem when there are two NSPs (called NSP 1 and NSP 2) and one customer. We assume that the customer sends a request at each decision epoch and that both NSPs propose an SLA (or nothing). If there is only one proposition, then the customer selects this proposition. If there are two offers then the customer chooses among these propositions. Thus, only one offer can be selected at each decision epoch. If NSP 1 is selected then NSP 2 does not provision any new SLA, and vice versa. If both offered SLAs are equivalent (regarding the customer's utility), the customer randomly selects one of them with uniform probabilities.

5.1 Optimal strategy when the NSPs collaborate

Here both NSPs are considered as a single one and the goal is to maximize the sum of their rewards. The approach to compute the optimal reward and the optimal strategies is the same as in Section 4.2.

Let $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) be the set of reachable states of NSP 1 (resp. NSP 2) as defined in Section 4.1. And let $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) be the transition function of NSP 1 (resp. NSP 2) as defined in Section 4.2. For each state $s_i^j = (b_i^1, \ldots, b_i^\Delta) \in \mathcal{S}_i$ (where $i \in \{1, 2\}$), we define $suc_\emptyset(s_i^j)$ as $suc_\emptyset(s_i^j) = (b_i^2, b_i^3, \ldots, b_i^\Delta, 0)$. The state $suc_\emptyset(s_i^j)$ is the successor of the state s_i^j when NSP i does not provision any SLA at the current decision epoch. It is clear that if $s_i^j \in \mathcal{S}_i$ then $suc_\emptyset(s_i^j) \in \mathcal{S}_i$ (because the used capacity corresponding to $suc_\emptyset(s_i^j)$ is less than the used capacity corresponding to s_i^j).

The set of states $\mathcal{S}$ of the whole system is included the Cartesian product $\mathcal{S}_1 \times \mathcal{S}_2$. If $s = (s_1, s_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ and $s' = (s'_1, s'_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ then there is a transition from s to s' iff at least one of those three cases occurs:

1. There is a transition from s_1 to s'_1 and $s'_2 = suc_\emptyset(s_2)$
2. there is a transition from s_2 to s'_2 and $s'_1 = suc_\emptyset(s_1)$
3. $s'_1 = suc_\emptyset(s_1)$ and $s'_2 = suc_\emptyset(s_2)$.

Note that case 3. is redundant, as $(s_i, suc_\emptyset(s_i))$ is always a transition of $\mathcal{T}_i$. The state space $\mathcal{S}$ is exactly the states in the strongly connected component to whom belongs the state $((0, \ldots, 0), (0, \ldots, 0))$ in the transition function $\mathcal{T}$.

Moreover, the transition function $\mathcal{T}$ over $\mathcal{S}$ is defined as:

$$\mathcal{T}(s, s') = \mathcal{T}_1(s_1, s'_1) + \mathcal{T}_2(s_2, s'_2). \tag{3}$$

Similarly to what has been done in Section 4.2, the maximum average reward of the whole system is given by:

$$\mathbb{E}(v_{\max}) = \max_{k=1..|\mathcal{S}|} \frac{\max_{s\in\mathcal{S}}(\mathcal{T}^k)(s, s)}{k}.$$

PROPOSITION 1. *Let $\mathbb{E}(v^1_{\max})$ (resp. $\mathbb{E}(v^2_{\max})$) be the maximum average reward of NSP 1 (resp. NSP 2) in case of one NSP, one customer. Then:*

$$\max(\mathbb{E}(v^1_{\max}), \mathbb{E}(v^2_{\max})) \leq \mathbb{E}(v_{\max}) \leq \mathbb{E}(v^1_{\max}) + \mathbb{E}(v^2_{\max}).$$

SKETCH OF PROOF. For $i \in \{1,2\}$, set $s_i^0 = (0,\ldots,0)$. The inequality $\mathbb{E}(v_{\max}^1) \leq \mathbb{E}(v_{\max})$ comes from the fact that $\mathcal{S}_1 \times \{s_2^0\} \subseteq \mathcal{S}$ and that if $\mathcal{T}_1(s_1^k, s_1^j) \neq -\infty$, then by construction $\mathcal{T}((s_1^k, s_2^0),(s_1^j, s_2^0)) = \mathcal{T}_1(s_1^k, s_1^j) + \mathcal{T}_2(s_2^0, s_2^0) = \mathcal{T}_1(s_1^k, s_1^j)$. As a consequence, for each cycle of $\mathcal{T}_1$, there is a cycle of same length and same weight in $\mathcal{T}$ and $\mathbb{E}(v_{\max}^1) \leq \mathbb{E}(v_{\max})$. By symmetry argument, exchanging the roles of $\mathbb{E}(v_{\max}^1)$ and $\mathbb{E}(v_{\max}^2)$ is enough to prove that $\mathbb{E}(v_{\max}^2) \leq \mathbb{E}(v_{\max})$ and the left inequality.

The right inequality is a direct consequence of the fact that $\mathcal{S} \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ and Equation (3): from each cycle of $\mathcal{T}$ one can find two cycles (a) with the same length respectively in $\mathcal{T}_1$ and $\mathcal{T}_2$, (b) whose weight is the sum of the weights of those two cycles. As a consequence, the mean weight of this cycle in $\mathcal{T}$ is the sum of the mean weight of a cycle in $\mathcal{T}_1$ and of a cycle in $\mathcal{T}_2$. □

5.2 Optimal strategy when the NSPs are selfish

There are two possible cases when the NSPs are selfish. First, each NSP has a dominant strategy, i.e., for each NSP, there is an SLA that maximizes its expected average reward, regardless of the NSP used capacity rate or the NSP state. Second, one NSP (or both NSPs) has no dominant strategy. These two cases lead to a slightly different models but do not impact the results.

5.2.1 Dominant Strategy

We assume here that each NSP has a dominant strategy corresponding to offer one SLA. If such an SLA exists, then it is clearly the one that maximizes the customer's utility. Thus, offering this SLA at each decision epoch is a dominant strategy for both NSPs. Let q_1^* (resp. q_2^*) be this SLA for NSP 1 (resp. NSP 2). The configuration where each NSP i offers q_i^* is a Nash equilibrium of the game [13][2].

If the utility of the customer when accepting q_1^* (resp. q_2^*) is greater than its utility when accepting q_2^* (resp q_1^*), the customer will always select the offer of NSP 1 (resp. NSP 2). In this case, the dominant NSP will monopolize the market.

If the customer's utility is the same for q_1^* and q_2^*, we assume that the customer selects randomly (with probability 1/2) between q_1^* and q_2^*. The different states of the system and the transitions between them can be modeled as a Markov chain. A state of the Markov chain is a pair $[s_1, s_2]$ where $s_1 \in \mathcal{S}_1$ and $s_2 \in \mathcal{S}_2$. Let $suc_\emptyset(s_i)$ be the successor of s_i as defined in Section 5.1. For each $s_i = (b_i^1, \ldots, b_i^\Delta)$, we define $suc_{\mathrm{q}}(s_i)$ as $suc_{\mathrm{q}}(s_i) = (b_i^2, \ldots, b_i^\Delta, b_i^*)$, where b_i^* is the bandwidth of q_i^*. In other words, $suc_\emptyset(s_i)$ is the successor of s_i when NSP i does not provision any SLA at the current time and $suc_{\mathrm{q}}(s_i)$ is the successor of s_i when NSP i provisions q_i^* at the current time.

All the recurrent states in the Markov chain are in the form $[(b_1^1, \ldots, b_1^\Delta), (b_2^1, \ldots, b_2^\Delta)]$ where $b_1^k = b_1^*$ and $b_2^k = 0$ or $b_1^k = 0$ and $b_2^k = b_2^*$, because each NSP offers its best (dominant strategy) SLA. Thus, at each decision epoch, the customer selects surely one of them. It follows that the number of recurrent states of the Markov chain is 2^Δ.

There is a transition (with probability 1/2) from state $s = [s_1, s_2]$ to $s' = [s_1', s_2']$ iff:

$$\begin{cases} s_1' = suc_\emptyset(s_1) \text{ and } s_2' = suc_{\mathrm{q}}(s_2) \\ \text{or} \\ s_1' = suc_{\mathrm{q}}(s_1) \text{ and } s_2' = suc_\emptyset(s_2) \end{cases}$$

note that the Price of Anarchy can be arbitrary large depending on the price difference between the Nash equilibrium strategy SLA (q_i^*) and the optimal SLAs that maximize the NSP reward if there is only one NSP. Example 2 provides an instance where the Price of Anarchy is larger than 1.

Example 2. In this example, we assume that NSP 1 and NSP 2 have the same maximal capacity $c_{\max}$ = 100Gbps and the same set of SLAs: q^1 with bandwidth b^1 = 30Gbps= $30\% \cdot c_{\max}$ and q^2 with bandwidth b^2 = 50Gbps= $50\% \cdot c_{\max}$. The duration of all SLAs is $\Delta = 2$ units of time and the failure function is $f(\rho) = \rho$. Suppose that the request of the customer is such that the customer's utility is maximized by q^1. Thus, offering q^1 is a dominant strategy for both NSPs and the situation where both NSP 1 and NSP 2 offer q^1 is a Nash equilibrium. If the used capacity of an NSP is either $c_t = 0$ or $c_t = 30\% \cdot c_{\max}$, offering q^1 instead of nothing increases its expected reward (the NSP expected reward when its used capacity is at $c_t = 60\% \cdot c_{\max}$ are greater than its expected reward when its used capacity is at $c_t = 30\% \cdot c_{\max}$). We assume that when both NSPs offer the same SLA, then the customer selects randomly one of them with probability 1/2.

It is clear that the optimal strategy for both NSPs is to offer alternatively q^2 and nothing (NSP 1 offers q^2 at decision epoch $t = 0$ then NSP 2 offers q^2 at $t = 2$, etc.). The expected average reward for each NSP is then $\mathbb{E}(v_{\max}) = 25$ m.u. per decision epoch.

Computing the expected average reward in the situation of Nash equilibrium described above can be done as follows: Since both NSPs offer q^1 at each decision epoch, and since the customer selects randomly one of them at each decision epoch, the states of the system can be modeled as a Markov chain. Fig. 4 shows the Markov chain associated to the system when both NSPs play their Nash equilibrium strategy. The states are combinations of two single NSP states and the arrows represent the transitions between the states. Each transition is shown with its probability. The recurrent states are those in the dotted rectangle. The transition matrix of the reduced Markov chain is given by:

$$\begin{array}{c} \\ s^3 \\ s^4 \\ s^5 \\ s^6 \end{array} \begin{array}{c} \begin{array}{cccc} s^3 & s^4 & s^5 & s^6 \end{array} \\ \begin{pmatrix} 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \end{pmatrix} \end{array}$$

And the stationary distribution is $(1/4, 1/4, 1/4, 1/4)$. Thus,

[2]A configuration where each player plays its dominant strategy is always a Nash equilibrium [14].

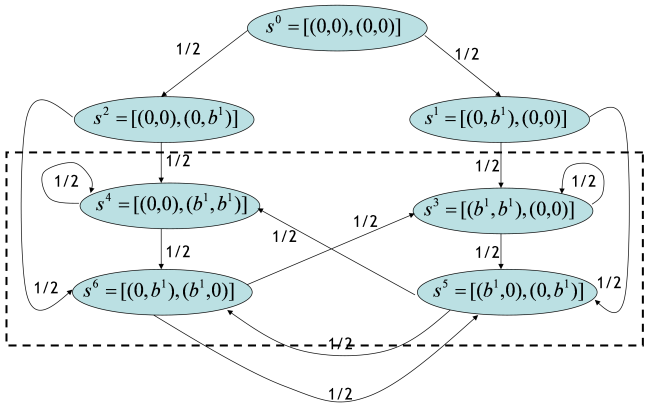

Figure 4: The Markov chain associated to the NSPs in Example 2. The recurrent states are in the dotted rectangle.

the expected average reward of each NSP is:

$$\mathbb{E}(v_{\max}) = \frac{1}{4}60(1 - f(0.6)) + \frac{2}{4}30(1 - f(0.3)) = 16.5$$

Thus, the Price of Anarchy in this case is $25/16.5 \approx 1.51$.

PROPOSITION 2. *The Price of Anarchy can be arbitrary large depending on the SLAs of the NSPs.*

SKETCH OF PROOF. As shown in Example 2, the Nash equilibrium strategy of the NSPs can differ from the optimal strategy if they collaborate. It is easy to design a game where the revenues gathered by the optimal strategy are arbitrary larger than the revenues gathered by the Nash equilibrium strategy. For instance, it is sufficient to consider the same parameters as in Example 2 and to set the bandwidth of q^1 as small as wanted. This gives a Price of anarchy as large as wanted. □

5.2.2 Absence of dominant strategy

Depending on its state, an NSP may proposes nothing instead of the closest SLA to the customer's request (for example, when the NSP has a too high used capacity). This case slightly impacts the structure of the Markov chain as described in Section 5.2.1 because it allows that both NSPs offer nothing to the customer. This leads to having transitions from a state $s = [s_1, s_2]$ to $s' = [s'_1, s'_2]$ where $s'_1 = suc_\emptyset(s_1)$ and $s'_2 = suc_\emptyset(s_2)$. However, the approach to compute the equilibrium expected reward remains the same. In this case too, it is possible to have instances where the Price of Anarchy is unbounded.

6. COMPETITION WITH REPUTATION MECHANISM

All the results stated in the previous sections are based on the assumption that the customers are only sensitive to the QoS parameters and the price of the SLAs. However, the customer is also sensitive to past failures he experimented with the NSP, modeled as its NSP reputation. Unfortunately, with a dynamic reputation (evolving during time), the Markov chain defined in section 5 is non-homogeneous. Section 6.2 explains this non-homogeneity thus motivating an evaluation by simulation.

We assume that all the customers are aware of the NSP reputation. We do not address the reputation propagation mechanism among the customers. The interested reader can refer to chapters 23 and 27 of [15] for an overview of reputation systems.

6.1 Computing the reputation value

We simply consider the reputation of an NSP as the (statistical) probability of success of its provisioned SLAs:

$$r_t(i) = 1 - \frac{\#fail(i)}{\#select(i)}$$

where $\#fail(i)$ is the number of SLAs provisioned by NSP i that were violated and $\#select(i)$ is the total number of SLAs provisioned[3] by NSP i.

6.2 Customer's choice regarding the NSP reputation

The reputation of an NSP can be seen as a probability of success of the SLA it offers. Thus, the utility of the customer according to an offer can be weighted by its reputation. More formally:

$$\Pr[select\ q_1^*] = \frac{r_t(1)U(q_1^*)}{r_t(1)U(q_1^*) + r_t(2)U(q_2^*)}$$

where q_1^* (resp. q_2^*) is the SLA proposed by NSP 1 (resp. NSP 2). The same formula applies for NSP 2 (by replacing q_1^* by q_2^*).

Thus, the transition probabilities between the states of the system are dependent upon the NSP reputation, which evolves at each decision epoch. This implies that the Markov chain associated to the system (as described in Section 5.2) is non-homogeneous. Thus, it is difficult to obtain analytical results and to compute the optimal strategies that maximize the rewards of the NSPs. This motivates the use of learning algorithms as described in the following section.

7. LEARNING FRAMEWORK

This second part of this paper evaluates if learning algorithms can learn a strategy capturing both the network capacity state and the NSP reputation, and if they find a trade-off between gathering high revenues and keeping good reputation. In this section, we provide an extension of the learning framework of [3, 5] to support the time durations of SLAs which plays a key role on the network capacity state.

7.1 Learning algorithms

In this section, we propose an adaptation of two learning algorithms: the Linear Reward Inaction (LRI) and the Q-Learning. Learning algorithms follow a scheme of 3 primitives : **select**, **observe**, **update**. The **select** primitive applies the learning algorithm policy to determine an optimal strategy. The **observe** primitive is a phase where the algorithm waits and observes the environment response (defined by the **reward**) to its proposed strategy. The **update** primitive modifies the knowledge used to select the strategy according to the observed reward.

[3]Recall that only the offers accepted by the customer are provisioned.

7.1.1 Linear Reward Inaction Algorithm

The LRI maintains a probability vector π_t where each $\pi_t(j)$ is the probability to offer q_i^j at decision epoch t to the customer. If the action succeeds (i.e., the offered SLA is selected by the customer), then the probability vector is updated as follows:

$$\pi_{t+1}(j) = \pi_t(j)\left(1 - \beta\frac{v_t}{v_{\max}}\right) + \beta\frac{v_t}{v_{\max}}$$

$$\pi_{t+1}(k) = \pi_t(k)\left(1 - \beta \cdot \frac{v_t}{v_{\max}}\right) \quad \forall k \neq j$$

where q_i^j is the selected SLA, q_i^k denotes all the other SLAs, v_t is the reward at time t (the price of q_i^j) and β is a learning parameter s.t. $0 < \beta < 1$. Note that the LRI algorithm is stateless, it does not take into account the NSP state (remaining capacity, etc.) in the decision process.

7.2 Q-Learning Algorithm

7.2.1 Markov Decision Process

The Q-Learning algorithm performs on a Markov Decision Process (MDP) model. An MDP is formalized as a 4-tuple $\{S, A, P, R\}$ where:

- S is the set of states of the MDP. The structure of a state is explained in the following section,
- A is the set of actions. It corresponds to the set of SLAs of the NSP,
- $P(\sigma, j, \sigma')$ is the probability to reach the state σ' when being in the state σ and offering the SLA q_i^j,
- $R(\sigma, j, \sigma')$ is the reward when reaching the state σ' after being in the state σ and offering the SLA q_i^j. It corresponds to the price of q_i^j if q_i^j is selected by the customer, 0 otherwise.

7.2.2 Capacity grade and MDP states

The used capacity rate should be in the MDP states so that the Q-Learning algorithm can learn strategies taking into account the remaining capacity of the NSP. However, the used capacity rate should be discretized in order to avoid combinatorial explosion of the number of states. Thus, it is converted into capacity grade (denoted by G). An example of discretization can be seen in Table 1. Recall that the failure probability of an SLA is correlated with the used capacity rate of the NSP. Table 1 shows the capacity grade corresponding to each interval of used capacity rate, and the associated failure probability. These values are those used to perform simulations in Section 8.

A state σ of the MDP is a pair (req, G) where req is a request profile and G the capacity grade of the NSP. If there is a single customer, then the request does not matter since we assume that a customer always sends the same requests. However, if there are more than one customer, then putting the request profile in the state allows the NSP to adapt its strategy depending on the customer profile.

Capacity grade G	Used Capacity Rate	Failure probability $f(\rho)$
0	$\rho = 1$	1
1	$0.95 \leq \rho < 1$	0.99
2	$0.90 \leq \rho < 0.95$	0.90
3	$0.80 \leq \rho < 0.90$	0.80
4	$0.60 \leq \rho < 0.80$	0.50
5	$0.30 \leq \rho < 0.60$	0.20
6	$\rho < 0.30$	0.05

Table 1: Relation between capacity grade, used capacity rate and SLA violation probability.

7.2.3 The Q-Learning algorithm

The Q-Learning [16] is a *model-free* algorithm, i.e., it does not need to know the transition probabilities of the MDP to perform. It is a useful propriety since the transition probabilities are dependent upon the customer's response, the failure probability and the other NSP offers. These values are not known by the NSP.

The Q-Learning algorithms compute the optimal *Q-values* of each pair (state, action) of the MDP at each decision epoch t. The Q-value of a pair (σ, a) at decision epoch t is defined as $Q_t(\sigma, a) = \mathbb{E}[R_t \mid \sigma_t = \sigma, a_t = a]$ where $R_t = \sum_{k=0}^{\infty} \gamma^k v_{t+k}$, with v_t is the expected reward at decision epoch t and $\gamma \in (0, 1]$ is a discount factor. Thus, $Q_t(\sigma, a)$ is the expected discounted reward when being in state σ and performing action a at decision epoch t.

The Q-Learning algorithm first initializes all the variable values, then, at each decision epoch, selects an SLA to offer according to some Q-based policy. The most common policy is the ϵ-greedy one. It consists, when being in state σ, to select the action a having the highest Q-value $Q_t(\sigma, a)$ with probability $1 - \epsilon$, and selecting another action at random with probability ϵ for some positive $\epsilon \ll 1$. After observing the customer's response (and thus getting v_t and the new state σ_{t+1}), the Q-Learning algorithm updates the Q-values according to formula (4), where α_t is a learning rate initialized with a value in $(0, 1)$, then decreased.

$$Q_{t+1}(\sigma, a) \leftarrow (1 - \alpha_t)Q_t(\sigma, a) + \alpha_t(v_t + \gamma \max_{a' \in A} Q_t(\sigma_{t+1}, a')) \quad (4)$$

Algorithm 1 Q-Learning algorithm

```
Initialization
loop
  At each decision epoch t
    Select an SLA a_t = q_i^j according to ε-greedy policy
    Observe reward v_t and new state σ_{t+1}
    Update the Q-values according to formula (4)
end loop
```

7.3 Risk-aware Learning framework

The learning framework **select**, **observe**, **update** is particularly impacted when taking into account the fact that SLA violations might occur during a long time-period. This forces to shift the reward observation until the end of the SLA duration.

Algorithm 2 Temporal learning framework

```
while Request at decision epoch t do
  Select SLA w.r.t. customer request
  Propose SLA to customer and Observe response
  if customer response is positive then
    provision the SLA
  end if
  for each SLA still provisioned at t do
    if there is a failure at t then
      Refund customer: Observe a negative reward
    else
      Observe a positive reward
    end if
  end for
  Update learning data
  for each SLA ending at t do
    Update network capacity grade (release resources)
  end for
end while
```

Algorithm 2 shows the temporal learning framework as it is impacted by the temporal constraints. At a request arrival, the NSP selects an SLA according to some policy and observes the customer's choice. It provisions the SLA if it is chosen by the customer. Then it observes if a failure occurs (which is dependent upon its current resource state). In case of failure (i.e., SLA violation), it refunds all the customers whose SLAs are still provisioned. Otherwise it collects the SLA prices for decision epoch t. Finally, the SLAs provisioned at decision epoch $t - \Delta$ (if any) are released and the learning data are updated.

8. SIMULATION RESULTS

In order to evaluate the behavior of learning algorithms (LRI and Q-Learning) and their ability to learn reputation-aware and risk-aware strategies, we perform simulations with two NSPs and one customer, then with two NSPs and two customers.

8.1 Simulated algorithms

The Q-Learning and LRI algorithms are compared to two "trivial" strategies:

- **Min strategy:** This strategy consists in offering always the cheapest SLA that satisfies the customer's requirements.
- **Uniform strategy:** It consists in offering at random and with uniform probability one of the SLAs that satisfy the customer's requirements.

8.2 Simulations with 1 customer and 2 NSPs

8.2.1 Simulation settings

We present the simulation results performed on the topology presented in Fig. 1. Each NSP has a set of 4 SLAs as described in Table 2. On table 2, SLAs are presented in an increasing price order.

The SLA duration $\Delta = 40$ time units and each NSP has a capacity of $10,000$Mbps. The customer request has the same QoS parameters as q_i^1, except for the delay which is 30ms. Thus, all the SLAs satisfy the customer's requirements. Considering the learning parameters, we set $\beta = 0.02$ for the LRI algorithm and $\alpha_0 = 0.5$ and $\gamma = 0.8$ for the Q-Learning algorithm.

SLA	Bandwidth	Delay	Packet-loss
q_i^1	100Mbps	20ms	0.1%
q_i^1	200Mbps	20ms	0.1%
q_i^2	250Mbps	30ms	0.1%
q_i^3	2500Mbps	5ms	0.001%

Table 2: Set of SLAs of NSP i ($i = 1, 2$).

Each NSP uses a particular algorithm or strategy, denoted *"Algo1 vs Algo2"* on the figures; *Algo1* is used by the NSP for which the results are exhibited by the curves, and *Algo2* is the opponent NSP's strategy. *QL* stands for Q-Learning, *Min* for the Min strategy and *Uniform* for the Uniform strategy. When both NSPs use the same algorithm, it is simply denoted by *"Algo"* instead of *"Algo1 vs Algo2"*.

Simulations were performed during 10000 rounds, where each round is repeated 100 times. All the results have been averaged according to the 100 repetitions.

8.2.2 Results

Fig. 5 shows the average reward **per request** of the first NSP when using different algorithms. . The results show that, after a short learning phase, the rewards stabilize, except those of the LRI algorithm which continue to slowly increase. The highest rewards are obtained when NSP 1 uses the LRI algorithm against the opponent NSP using the uniform strategy, followed by NSP 1 using Q-Learning algorithm against the opponent NSP using the uniform strategy. It seems that the min strategy performs bad against all other strategies, which means that the customer prefers to be upgraded with the current simulation parameters. The LRI algorithm perform slightly better than the Q-Learning algorithm, and the learning algorithms perform better than the trivial strategies.

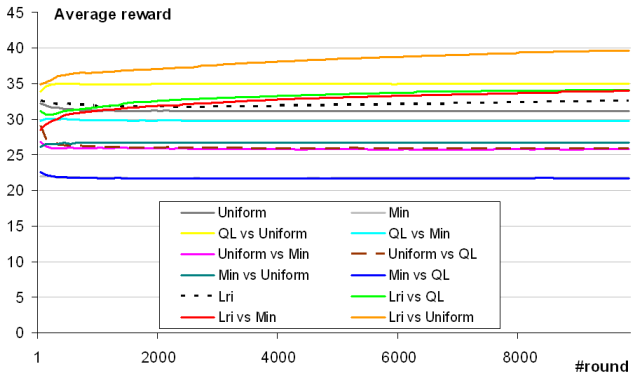

Figure 5: NSP rewards

Fig. 6 shows the average reputation of NSP 1. Obviously, the min strategy keeps the highest reputation against all other strategies. Since the min strategy offers always the

cheapest SLA (which is also the SLA with the lowest QoS parameters), it minimizes the NSP used capacity and thus minimizes also the SLA failure probability, which leads to a high reputation. All other strategies stabilize between a reputation of 0.8 and 0.9, which seems a good trade-off between a high reputation and high immediate rewards.

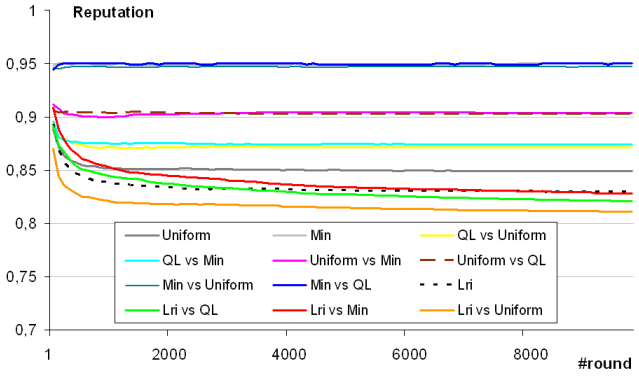

Figure 6: NSP reputation

Fig. 7 shows the average used capacity of NSP 1. The more the capacity is used, the higher are the rewards. However, all the strategies maintain the used capacity between 20% and 45%, except the min strategy that maintains the used capacity less than 20%.

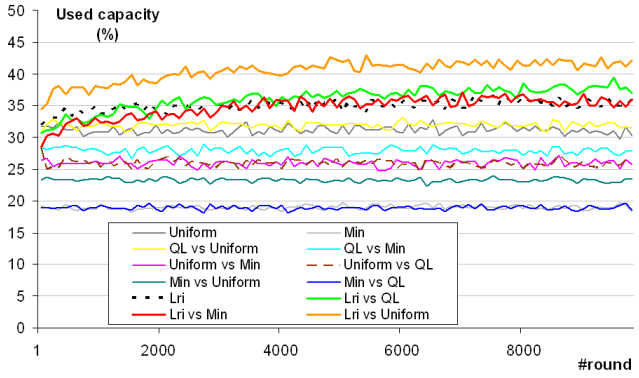

Figure 7: NSP used capacity (%)

8.3 Simulations with 2 customers and 2 NSPs

This section exhibits the simulation results with 2 customers and 2 NSPs. The setting parameters are the same as in Section 8.2.1, except that there are two customers sending their requests to both NSPs at the same time.

Fig. 8 shows the **per request** average rewards of NSP 1. The highest rewards are obtained when NSP 1 uses the LRI algorithm against the opponent NSP using the uniform strategy, as in the case of a single customer. It seems that the LRI algorithm outperforms the Q-Learning algorithm when facing simultaneous requests of 2 customers. The Q-Learning algorithm suffers from this configuration as it presents worse results than with 1 customer. The min strategy presents the worst results facing any of the other strategies.

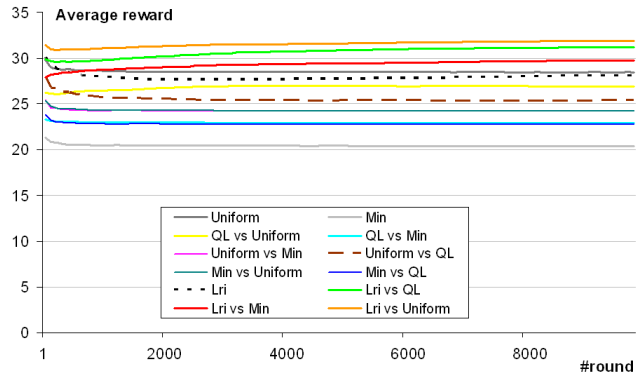

Figure 8: NSP average rewards

The evolution of the reputation value in this case is not presented because it follows the same trend as in the case of 1 customer. Fig. 9 shows the evolution of the NSP used capacity. The correlation between the used capacity and the rewards can be seen in the same way that in the case of 1 customer. It seems that the Q-Learning algorithm performs

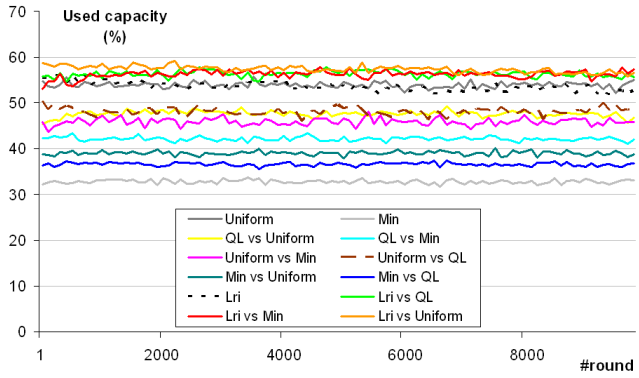

Figure 9: NSP used capacity

worse than the LRI algorithm. In both cases of 1 and 2 customers, the LRI algorithm performs well. These results are inconsistent with those of [3] (where the Q-Learning algorithm performs better than the LRI algorithm). But, the problem addressed in [3] did not capture the NSP reputation, nor the multiple SLA overlapping and had a different customer preference function.

9. CONCLUSION

Assuring QoS connectivity is a major challenge for current and future networks. We investigated the optimal strategy of an NSP in the SLA negotiation taking into account the risk of SLA violations and we presented a method to compute such a strategy for 1 or 2 NSPs and one customer. Then, we defined a model for the NSP reputation embedding the SLA duration. As the previous results do not hold when introducing the reputation, we proposed a learning framework tackling this more complex problem. Simulations showed that learning algorithms outperform the strategies of offering the cheapest SLA or an SLA at random. Surprisingly, the LRI algorithm surpasses the Q-Learning algorithm. This may be due to the faster convergence of the LRI

algorithm. As a future work, we plan to study algorithm performance under various arrival laws, and how these different laws impact the NSP optimal strategy.

10. REFERENCES

[1] Michael P. Howarth and al. End-to-end quality of service provisioning through inter-provider traffic engineering. *Computer Communications*, 29(6), 2006.

[2] Tristan Groléat and Hélia Pouyllau. Distributed inter-domain SLA negotiation using Reinforcement Learning. In *Integrated Network Management*, 2011.

[3] Mohamed Lamine Lamali, Hélia Pouyllau, and Dominique Barth. SLA negotiation: Experimental observations on learning policies. In *VALUETOOLS*, pages 243–252, 2012.

[4] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: utility functions, random losses and ECN marks. *Networking, IEEE/ACM Transactions on*, 11(5):689 – 702, oct. 2003.

[5] Mohamed Lamine Lamali, Dominique Barth, and Johanne Cohen. Reputation-Aware Learning for SLA Negotiation. In *Networking Workshops*, pages 80–88, 2012.

[6] Panita Pongpaibool and Hyong S. Kim. Providing end-to-end service level agreements across multiple ISP networks. *Comput. Netw.*, 46(1):3–18, September 2004.

[7] Kalika Suksomboon, Panita Pongpaibool, and Chaodit Aswakul. An equilibrium policy for providing end-to-end service level agreements in interdomain network. In *WCNC*, 2008.

[8] Ming Xia, Massimo Tornatore, Charles U. Martel, and Biswanath Mukherjee. Risk-aware provisioning for optical WDM mesh networks. *IEEE/ACM Trans. Netw.*, 19(3):921–931, 2011.

[9] Axel Keller & al. Quality assurance of grid service provisioning by risk aware managing of resource failures. In *CRiSIS*, 2008.

[10] Dominic Battré & al. Assessgrid strategies for provider ranking mechanisms in risk-aware grid systems. In *GECON*, 2008.

[11] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *Synchronization and Linearity: an algebra for discrete event systems*. 1992.

[12] Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, September 1978.

[13] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[14] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*, volume 1 of *MIT Press Books*. The MIT Press, 1994.

[15] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[16] Christopher J. C. H. Watkins and Peter Dayan. Technical Note Q-Learning. *Machine Learning*, 8, 1992.