e-Auctions for Multi-Cloud Service Provisioning

Marco Anisetti, Claudio A. Ardagna, Ernesto Damiani
DI – Università degli Studi di Milano
Crema (CR), 26013, Italy
Email: firstname.lastname@unimi.it

Piero A. Bonatti, Marco Faella, Clemente Galdi, and Luigi Sauro
Università di Napoli "Federico II"
Napoli (NA), 80126, Italy
Email: firstname.lastname@unina.it

***Abstract*—The cloud computing paradigm requires solutions supporting customers in the selection of services that satisfy their functional and non-functional requirements. These solutions must *i)* support the dynamic, multi-cloud nature of service provisioning, *ii)* manage scenarios where no total preference relation over service properties is available, and *iii)* prevent providers from misrepresenting or overstating their properties.**

In this paper we put forward the idea of modeling multi-cloud provisioning scenarios as procurement e-auctions (where the auctioneer is the customer and the bidders are service providers). We introduce a service selection process based on matching and ranking algorithms, and an e-auction mechanism that addresses the above requirements, encouraging trustworthy bids and therefore improving the truthfulness on the e-auction outcome. Finally we describe the implementation of a prototype used to evaluate the performance of our approach with respect to traditional query-based engines.

***Keywords*-Cloud computing, e-Auction, Services, Truthfulness.**

I. Introduction

The cloud computing paradigm is becoming the preferred solution for service provisioning and procurement. It supports a vision of remote IT where hardware and software resources are distributed as commodities over the Internet and used on a *pay-as-you-go* basis [1]. A variety of services are developed and made accessible through the cloud by different service providers. Resource provisioning is often regulated by service-level agreements (SLAs), that is, contracts between customers and cloud/service providers that specify both functional and non-functional agreements. The multi-cloud nature of service provisioning—where multiple providers offer their resources to customers on different cloud infrastructures—calls for solutions to support customers in selecting cloud services that satisfy their functional and non-functional requirements.

In this paper, we argue that *choosing* a service (and a provider) that optimally satisfies those requirements is not the only issue involved. It should be taken into account that various factors (to name but a few: suppliers' profit-orientation, uneffectiveness of customer protection, lack of penalty for service misrepresentation) may induce providers to offer contracts that are suboptimal for the customer (e.g., with higher price, or perhaps lower QoS to save resources for possible future customers). Service selection mechanisms may be more ambitious, and exploit the competition between providers to obtain better SLAs for the customer, by incentivizing providers to offer the best contracts they can fulfill.

We pursue this goal by modeling multi-cloud provisioning scenarios as procurement e-auctions (*e-auctions* in the following), where the auctioneer is the buyer and the bidders are the sellers. The bids usually specify the price of the goods or services sold by the bidders. In our approach, automated SLA negotiation mechanisms are replaced by an e-auction: the customer sends a request for a service/application to an agent acting as an auctioneer. Interested application service providers (ASPs) submit their bids to the auctioneer for the service request being auctioned. The auction is awarded to one of the providers offering some of the best contracts. We note that it is not unusual for the same provider to offer multiple bids for the same service, as in the case of solutions featuring different service bundles [2].

An important aspect to consider here is the mechanism used to select the contract to be subscribed. For example, if the contract is always exactly one of the best bids, then auction theory [3] shows that, in some contexts, providers are incentivized to make offers that increase their own profit, which usually has negative effects from a customer point of view and on the overall quality of the service. The standard solution to this problem, in microeconomics, consists in decoupling the winner's offer and the final contract as in *second-price* procurement auctions, where the auctioned good is bought at the second-best price (the lowest price higher than the winner's offer). In practice, the distance between the optimal offer and the second-best offer is usually moderate; moreover, in second-price auctions, the best strategy for all bidders is always offering their best possible price (a property known as *truthfulness*), that is, providers have no incentive to offer the service at higher price. However, in the scenarios of our interest, bids are more complex than prices (Section II): they consist in complex contracts (SLAs) that specify functional as well as non-functional properties, possibly including QoS, security features, and many other properties, besides price. In this context, it is typically impossible to identify a *total* preference relation over the set of bids. For example, bids may specify alternative security

978-1-4799-5066-9/14 $31.00 © 2014 IEEE
DOI 10.1109/SCC.2014.14

mechanisms whose strength is not comparable [4]; finding an optimal tradeoff between different contract attributes (e.g., price vs. QoS, or performance vs. security) may be impossible due to incomplete knowledge. When a customer's preferences over possible outcomes are partial *i)* classical auctions are too rigid [5], *ii)* there exists no clear notion of "second price", and *iii)* winner selection becomes a delicate choice that may affect the guarantees on auction outcomes. Approximating the partial order with one of its linearizations introduces further problems and typically does not preserve the truthfulness property [5].

In this paper we address the above problems by introducing a service selection process based on a generalization of second price auctions that supports partial preferences and enjoys a weak form of truthfulness (Section IV). This property suffices to guarantee that the auction outcomes are at least as good as if the providers were all truthful (i.e., as if they offered the set of all possible contracts that they can actually fulfill without incurring a loss, including all optimal combinations of price, QoS, security features, and the like). The service selection process comprises the matching of requirements and offers (Section III), which are rich enough to cover the contracts of our interest. A prototype of our e-auction approach has been developed to evaluate its performance, and in turn its applicability to a cloud environment (Section V).

II. E-AUCTION SLAs

In this section, we define non-functional properties of interest and how they can be integrated within SLA contracts.

A. Non-functional properties of services

Substantial effort has been done to the aim of defining a taxonomy of non-functional properties for software and services (e.g., [6]–[8]). In this paper, we consider macro categories similar to the one specified in [6], and extend them to fit our e-auction scenario. These categories include among the others: *i) Security* evaluating the capability of a service to protect accesses to information and data from unauthorized parties, and to guarantee accesses from authorized services/applications (e.g., confidentiality, authenticity, and integrity); *ii) Performance* evaluating the performance of services (e.g., response time, throughput, and timeliness); *iii) Dependability* evaluating the ability to deliver services that can justifiably be trusted and to avoid service failures that are more frequent and severe than acceptable [7] (e.g., reliability, safety, availability).

Each category *cat* is populated by specific non-functional properties that are the target of the e-auction process in this paper and are defined as follows. A non-functional property p is a pair ($\hat{p}$,$Attr$), where $p.\hat{p}$ is an abstract non-functional property (i.e., a label from a shared controlled vocabulary, such as confidentiality, guaranteed bandwidth, responsiveness, reliability) and $p.Attr$ is a set of class attributes

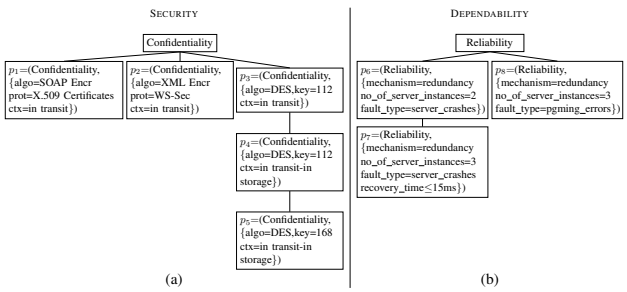


Figure 1. An example of a hierarchy of security (a) and dependability (b) properties

specifying the mechanisms implemented by the service to provide p. As an example, category *Security* contains property p=(*Confidentiality*,{`algo`=DES,`key`=112,`ctx`=in-transit}), meaning that the service providing this property guarantees confidentiality of data in transit by means of DES encryption algorithm with a 112-bit key.

A partial order can be defined over non-functional properties of each category, based on attribute values, inducing a set of hierarchies $\mathcal{H}_{cat}$ of properties (one for each category *cat*) as a pair ($\mathcal{P}$,$\preceq_P$), where $\mathcal{P}$ is the set of properties of category *cat* and $\preceq_P$ the partial order. Given two properties p_i and p_j in $\mathcal{P}$, we write $p_i \preceq_P p_j$ if p_i is weaker than p_j. These hierarchies are at the basis of our approach to e-auction for the selection of the best ASP offer. Figure 1 shows two examples of hierarchies for properties in categories *Security* and *Dependability*. For instance, let us consider the following non-functional properties of category *Security*: p_i=(*Confidentiality*,{`algo`=DES,`key`=112,`ctx`=in transit}) and p_j=(*Confidentiality*,{`algo`=DES,`key`=168,`ctx`=in transit-in storage}). According to our hierarchy in Figure 1(a), $p_i \preceq_P p_j$.

Moreover, we also consider other categories of non-functional properties that are relevant in the context of service provisioning and e-auction, such as, *service cost*. For instance, the same service can be released as *freeware*, providing basic functionalities and non-functional properties, or *under a license fee*, having more up-to-date functionalities and enhanced non-functional properties. This double release policy has been adopted by many commercial cloud-based services like *Dropbox*.

B. SLA Contracts

In a cloud environment, ASPs have to manage two contrasting needs. On one side, they aim to maintain the scalability of service provisioning; on the other side, the SLA contracts agreed with the client cannot be violated until they expire. As a consequence, the SLA contracts proposed by an ASP are influenced by self-adaptable cloud resource management mechanisms [9], especially for those properties insisting on resources which are limited in nature (e.g., guaranteed bandwidth). These resources become non-adaptable

resources for the ASP, and cannot be tuned to react to unpredictable events and workload fluctuations. Consequently, an ASP has an incentive to offer less resources than what is currently available, and tends to deliver less satisfactory services to clients. In this paper, we assume ASPs to be rational providers and use e-auctions to moderate the above trend, by introducing incentives to reach a tradeoff between provider profit and customer satisfaction. We propose a scenario where SLA contracts are stipulated with the help of an auctioneer service, acting as an intermediary between a customer that requests a service and the ASPs that offer services matching the customer's requirements.

The auctioneer collects the customer's requirements and the bids, selects an ASP based on the bids, and establishes the SLA contract with that provider. We note that, in our multi-cloud provisioning scenarios, the SLA contract should be continuously verified at runtime until it is valid. For simplicity, we assume here that SLA contract violations are visible to the client, so that any ASP who stipulates an offer that cannot be profitably fulfilled incurs a loss, due to high delivery costs, penalties for breaking the agreement, reputation damage, and so forth. In general, contract verification can be done by verifying validity of SLA contract properties via *monitoring* or *testing* techniques [10].

Customers define *SLA Preferences* over possible contracts, including requirements on functional and non-functional properties of services. ASPs instead specify *SLA Templates* that describe service offers (bids) in terms of those properties. Here, we consider non-functional properties only, although our e-auctions are general enough to handle functional requirements as well. SLA Preferences and Templates are defined as machine-readable documents. In abstract terms, they are n-tuples, where n is the number of specified non-functional properties p=($\hat{p}$,$Attr$). We note that, for each pair of properties p_i and p_j in a SLA Preference (SLA Template, resp.), $p_i.\hat{p} \neq p_j.\hat{p}$. SLA Templates also contain additional information such as *owner*, *contract validity*, and *remedies* in case of violations. Figure 2 shows an example of SLA Template, which considers property *Confidentiality* of category *Security* and property *Reliability* of category *Dependability*. After a bid is selected (winner bid), the corresponding SLA template is instantiated in a SLA contract to the agreement between the customer and the ASP. The selection of the auction winner relies on the ability to compare different SLA Templates associated to different bids including a set of non-functional properties.

```
SLA Template={
  owner=ASP1;
  validity=21-11-2014;
  Security:
    [Confidentiality,{algo=DES,key=112,ctx=in-transit}];
  Dependability:
    [Reliability,{mechanism=redundancy,
                  no_of_server_instances=2,
                  fault_type=server_crashes}]
}
```

Figure 2. An example of SLA Template.

III. Matching and Ranking of SLA Templates

The process of deciding the e-auction winner considers the service request by the auctioneer, including requirements on non-functional properties (SLA preference), and requires to order bids of the selected ASPs (SLA Templates) on the basis of the specified properties. While in some cases (e.g., service cost) this ordering is quite straightforward and results in a total order, in other cases (e.g., security-related properties) it is more complicated and could eventually lead to partial orders [11]. In this paper we adopt a priority-based solution, where each abstract property $p.\hat{p}$ is associated with a priority that identifies its importance in the ordering process. This approach still produces a partial preference order, because non-functional properties are themselves partially ordered (see Figure 1). We note that the auction mechanism in Section IV applies also to arbitrary partial orderings in which properties are not prioritized.

The ranking approach in this section is based on two subprocesses: *i)* a *matching process* that receives as input the SLA Preference of the customer and a set of SLA Templates of ASP services, and returns as output the SLA Templates that address requirements in the SLA Preference; *ii)* a *ranking process* that receives as input the SLA Templates selected by the matching process and returns as output an ordering of these templates based on priorities, thus identifying the e-auction winner.

A. Matching Process

The matching process considers the customer SLA Preference r=$\{p_1^r,\ldots,p_n^r\}$ and a set $\mathcal{T}$ of SLA Templates t_i=$\{p_1,\ldots,p_m\}$. We recall that each pair of properties p_j and p_k in r (t_i, resp.) has different abstract properties (i.e., $p_j.\hat{p} \neq p_k.\hat{p}$). We also note that each property requirement $p_j^r \in r$ can be of three types: *i)* a lower bound requirement $\underline{p_j^r}$, *ii)* an upper bound requirement $\overline{p_j^r}$, or *iii)* a range requirement $\underline{\overline{p_j^r}}$. Clearly, the range requirement is the conjunction of lower and upper bound requirements, denoted $[p_j^r,\overline{p_j^r}]$. The matching between r and a single t_i is successful if t_i satisfies r, as discussed in Definition 3.1; otherwise, t_i is discarded and not considered for the ranking process.

Definition 3.1 (Matching): Let r=$\{p_1^r,\ldots,p_n^r\}$ be a SLA Preference and t_i=$\{p_1,\ldots,p_m\}$ be a SLA Template. The matching process between r and t_i is successful iff $\forall p_j^r \in r$, $\exists p_k \in t_i$ *s.t. i)* $\underline{p_j^r} \preceq_P p_k$ (lower bound requirement), *ii)* $p_k \preceq_P \overline{p_j^r}$ (upper bound requirement), or *iii)* $p_k \preceq_P \overline{p_j^r} \wedge \underline{p_j^r} \preceq_P p_k$ (range requirement) based on hierarchies $\mathcal{H}_{cat}$.

The matching process applies to all templates in $\mathcal{T}$ and returns a set $A \subseteq \mathcal{T}$ of *admissible* templates, which represent the possible candidates for selection.

Table I
SET OF SLA TEMPLATES GIVEN AS INPUT TO THE COMPARISON PROCESS

t	Security	Reliability
t_1	(confidentiality,{algo=*DES*,key=168,ctx=*in transit-in storage*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=3, fault − type=*pgming-errors*})
t_2	(confidentiality,{algo=*DES*,key=168,ctx=*in transit-in storage*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=2, fault − type=*server-crashes*})
t_3	(confidentiality,{algo=*DES*,key=168,ctx=*in transit-in storage*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=2})
t_4	(confidentiality,{algo=*DES*,key=112,ctx=*in transit-in storage*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=5, fault − type=*server-crashes*})
t_5	(confidentiality,{algo=*DES*,key=112,ctx=*in transit-in storage*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=2, fault − type=*server-crashes*})
t_6	(confidentiality,{algo=*DES*,key=112,ctx=*in transit*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=3, fault − type=*pgming-errors*})
t_7	(confidentiality,{algo=*DES*,key=112,ctx=*in transit*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=2, fault − type=*server-crashes*, recovery − time<15ms})
t_8	(confidentiality,{algo=*SOAP Encr*, prot=*X.509 Certificates*,ctx=*in transit*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=2, fault − type=*server-crashes*, recovery − time<15ms})
t_9	(confidentiality,{algo=*SOAP Encr*, prot=*X.509 Certificates*,ctx=*in transit*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=3, fault − type=*server-crashes*, recovery − time<15ms})
t_{10}	(confidentiality,{algo=*XML Encr.*,prot=*WS-Sec*,ctx=*in transit*})	(Reliability, {mechanism=*redundancy*, no − of − server − instances=3, fault − type=*pgming-errors*})

B. Ranking Process

The ranking process produces a ranked order of templates $t \in A$ returned by the matching process. It generates the ranked list by comparing the properties in the templates according to their dominance hierarchy. To avoid inconsistencies in the ordering, we assume a *priority-based approach*, where a priority *pr* is assigned to each abstract property $\hat{p}$, as follows.

Definition 3.2 (Priority pr): Given the set of all abstract properties $\hat{p}_1,\ldots,\hat{p}_n$, a priority $pr \in \mathbb{N}^+$ is associated to each of them, such that $pr_i \neq pr_j$ with $i \neq j$. $pr_i < pr_j$ means that $\hat{p}_i$ with pr_i is applied first in the ranking process.

A priority is a positive integer pr_i, which establishes the order in which the properties in each template in A are evaluated to produce the ranking. The ranked list is then generated by iteratively applying one sub-process for each abstract property (e.g., Confidentiality, Reliability, Availability) according to their priority (e.g., pr_C=1, pr_R=2, pr_A=3). More in detail:

1) the sub-process managing the abstract property $\hat{p}$ with higher priority receives as input all admissible templates in A. Templates $t_i \in A$ are ordered on the basis of their property p_i=($\hat{p}$,*Attr*). Templates with the same property are further analyzed by the next (if any) sub-process to establish an order between them;
2) all subsequent sub-processes work on different subsets of templates, each one containing at least two templates still to be ordered. Templates with the same property are further analyzed by the next (if any) sub-process to establish an order between them.

Step 2 is repeated until either no templates need to be ordered or all abstract properties have been evaluated. We recall that, in every sub-process, templates are ordered by comparing properties p_i with the same abstract property $\hat{p}_i$ of category *cat*, on the basis of corresponding hierarchy $\mathcal{H}_{cat}$. We also note that the subsequent sub-process is selected on the basis of priority pr_i associated with each abstract property $\hat{p}_i$.

Example 3.1: Let us consider a SLA Preference requiring services with any type of confidentiality and reliability properties, and the SLA Templates in Table I. For conciseness, we assume templates specifying confidentiality and reliability properties only. Our matching process returns all templates $t_i \in A$ in Table I, with i=1,...10, and gives them as input to the ranking process. Let us assume, pr_C=1 and pr_R=2. The ranking process works as follows. SLA Templates in A (Table I) are first grouped by their confidentiality properties resulting in a set of equivalence classes $\{\{t_1,t_2,t_3\},\{t_4,t_5\},\{t_6,t_7\},\{t_8,t_9\},\{t_{10}\}\}$. Then, the five classes are ordered using corresponding $\mathcal{H}_{cat}$ of category Security, producing the partially ordered tree depicted in Figure 3 (first step). As an example, Figure 3 shows that every SLA Template in $\{t_6,t_7\}$ is weaker than every SLA Template in $\{t_4,t_5\}$ based on the security property, and in turn every SLA Template in $\{t_4,t_5\}$ is weaker than every SLA Template in $\{t_1,t_2,t_3\}$. Furthermore, $\{t_{10}\}$ and $\{t_8,t_9\}$ are not comparable with other SLA Templates. The second part of the ranking process then concentrates on classes $\{t_1,t_2,t_3\}$, $\{t_4,t_5\}$, $\{t_6,t_7\}$, and $\{t_8,t_9\}$. The reliability property in each of the above templates is used to order bids within each class. In particular, given the portion of the reliability property hierarchy in Figure 1, the final rank in Figure 3 is produced.

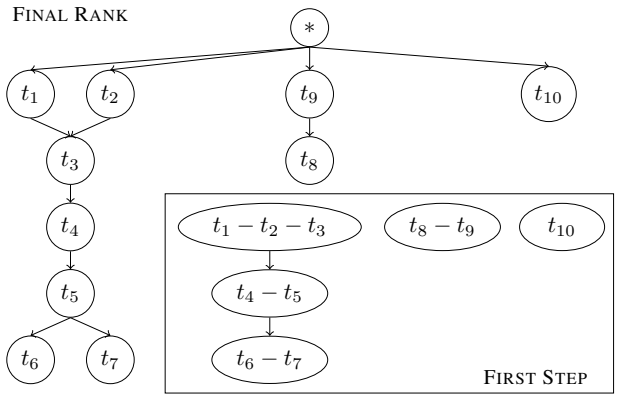


Figure 3. Ranking process of the SLA Templates in Table I; the inner square presents the first ranking step.

IV. FORMAL GUARANTEES ON SELECTED SLAS

In formal terms, the auction is specified as follows. First, the customer creates an auction by contacting the e-auction platform and submitting her requirements, in terms of: *i)* the set A^c of SLA Templates that are *admissible* for the customer (i.e., those that match the customer's request,

according to the matching algorithm specified in Section III), and *ii)* a (partial) preference relation $\leq^c$ over SLA Templates (i.e., the partial ordering[1] used by the ranking algorithm, based on the hierarchies $\mathcal{H}_{cat}$ and the customer's priorities, cf. Section III and Definition 3.2). When a is *strictly* preferred to b (i.e. $a \leq^c b$ and $b \not\leq^c a$), we write $a <^c b$.

In practice, A^c is identified compactly by means of a SLA preference, while $\leq^c$ can be specified by selecting pre-defined, property hierarchies (supplied by the e-auction platform) and setting the priorities over different properties.

Once the auction is started, each ASP i submits a bid $B_i \subseteq A^c$, (i.e. a set of SLA Templates matching the customer's requests). Let $\mathbf{B} = \langle B_1, \ldots, B_n \rangle$ denote the bid vector (where n is the number of bidders). The auction proceeds as follows:

1) For each provider i, let $\mathit{filter}(i, \mathbf{B})$ be the set of contracts in B_i that are *convenient* for the customer, in the sense that no other provider has made any strictly preferable offer. Formally,

$$\mathit{filter}(i, \mathbf{B}) = \{b \in B_i \mid \forall j \neq i, \forall a \in B_j\,, b \not<^c a\}.$$

The auctioneer then picks a provider w as the winner from those who submitted one of the top offers in the final rank, that is, (equivalently) the members of

$$cw(\mathbf{B}) = \{i \mid \mathit{filter}(i, \mathbf{B}) \neq \emptyset\}\,,$$

(cw stands for *candidate winners*). The winner is selected from $cw(\mathbf{B})$ with uniform probability:

$$pw(i, \mathbf{B}) = 1/|cw(\mathbf{B})|.$$

2) The auctioneer transmits to the winner w the set of templates $\mathit{filter}(w, \mathbf{B})$.
3) The winner replies with a single, freely chosen template $b_w \in \mathit{filter}(w, \mathbf{B})$.

At this point, the auction terminates and the service is acquired from provider w according to the agreement b_w.

Now the question is: *Which bids B_i are more convenient for an ASP i?* Roughly speaking, providers should better increase their chances of closing *better* contracts *from their perspective*. However, so far we have not modeled bidder preferences. For this purpose, for each bidder i we introduce a (possibly partial) preference relation $\leq_i$ over SLA Templates (and its strict version $<_i$). Moreover, each bidder i has a corresponding set of *admissible* SLA Templates A_i, which is meant to contain all and only the contracts that i can actually fulfill with some profit. Accordingly, for all templates $a \in A_i$ and $b \notin A_i$, $b <_i a$ holds (because by stipulating any $b \notin A_i$, bidder i incurs a loss). The best contracts i can obtain in context $\mathbf{B}$ are the members of

$$\max(i, \mathbf{B}) = \{b \in \mathit{filter}(i, \mathbf{B}) \mid \forall a \in \mathit{filter}(i, \mathbf{B}), b \not<_i a\}\,.$$

[1]In this paper, all preference relations are partial orders.

On paper bidders do not know the customer's preferences $\leq^c$, since they are not released to the bidders. However, some of these preferences are easily identifiable in practice by means of the hierarchies $\mathcal{H}_{cat}$. For example, from a customer's perspective, a SLA with lower price and higher QoS and security is better than any SLA with higher price and lower QoS and security. Of course, the customer may have additional preferences that are unknown to the bidders, concerning—say—the tradeoff between higher price and higher QoS, or higher performance and stronger security (such as those expressed by the priorities of Definition 3.2).

Interestingly, in our reference scenarios, bidders have opposite preferences on the above "obvious" cases. For instance, higher QoS and security consume computational resources that could alternatively be used for supporting more customers; so providers prefer contracts with higher price and lower QoS and security. Again, each provider may have additional preferences that apply to non-obvious cases.

We denote with $\leq$ (and its strict version $<$) the "obvious" customer preferences over entire SLA Templates induced by the property hierarchies $\mathcal{H}_{cat}$, and according to the above discussion we assume that:

$$a \leq b \;\Rightarrow\; a \leq^c b \qquad a < b \;\Rightarrow\; a <^c b \qquad (1)$$
$$a \leq b \;\Rightarrow\; b \leq_i a \qquad a < b \;\Rightarrow\; b <_i a \qquad (2)$$

Resuming our initial question, now we are ready to formalize the "obvious" preferences of a bidder i over alternative possible *bids* B_i. In the following, B_{-i} denotes the vector of bids submitted by all $j \neq i$, and (X, B_{-i}) denotes the bid vector obtained from $\mathbf{B}$ by replacing B_i with X. Finally, $\preceq_i$ denotes i's preferences over possible bids.

ASP Preference Postulate (APP): *For all $B', B'' \subseteq A^c$, $B' \preceq_i B''$ holds (at least) when for all $\leq^c$ satisfying (1), and for all B_{-i}, conditions 1, 2, and 3 below are satisfied:*

1) *if* $\mathit{filter}(i, (B'', B_{-i})) \cap A_i = \emptyset$ *then* $pw(i, (B', B_{-i})) \geq pw(i, (B'', B_{-i}))$*;*
2) *if* $\mathit{filter}(i, (B'', B_{-i})) \cap A_i \neq \emptyset$ *then* $pw(i, (B', B_{-i})) \leq pw(i, (B'', B_{-i}))$*;*
3) *for all* $b' \in \max(i, (B', B_{-i}))$ *there exists* $b'' \in \max(i, (B'', B_{-i}))$ *such that* $b' \leq_i b''$.

Moreover, $B' \prec_i B''$ holds (at least) when some of the disequalities $\geq$, $\leq$, and $\leq_i$ in the above conditions are strict for at least one relation $\leq^c$ and one vector B_{-i}.[2]

The APP simply says that surely B'' is at least as good as B' when—in all possible contexts—B'' does not increase the risk of a loss (cond. 1), it possibly increases the chances of closing a profitable contract (cond. 2), and it may improve the contract itself (cond. 3).

[2]We remark that the above conditions do not constitute a complete definition of $\preceq_i$ and $\prec_i$; the APP sets only a lower bound to these relations. In many cases, no bid B_i is the best option in *all* contexts, so the "obvious" preferences captured by the APP may be refined depending—say—on the ASP's propensity to risk.

Some auctions enjoy the truthfulness property, that is, a rational bidder's best strategy is offering what the bidder is actually willing to pay, in all contexts. The equivalent in our setting is offering exactly all profitable contracts, that is, $B_i = A_i$. Usually, with partial auctioneer preferences, truthfulness does not hold (cf. [5]). However, the application scenarios that satisfy (1), (2) and APP enjoy a slightly weaker property that—from a customer viewpoint—provides at least the same contract quality as truthfulness.

Definition 4.1: A bid B_i is *weakly truthful* iff $B_i \supseteq A_i$. A bidder i is weakly truthful if B_i is.

A weakly truthful ASP offers all of its admissible contracts, and possibly some additional contracts that are not really convenient from its viewpoint (a strategy called *overbidding*), in order to increase the probability of winning. An overbidding winner does not incur a loss if $\mathit{filter}(B_w, B_{-w}) \cap A_w \neq \emptyset$ (so that w can pick a profitable contract $b_w \in A_w$). Regardless the strategic reasoning on overbidding and risk propensity, a rational bidder's optimal strategy is always weakly truthful:

Theorem 4.1: Suppose that B'_i, is not weakly truthful. Then there exists a weakly truthful B''_i such that $B'_i \prec_i B''_i$.

Proof: By assumption, there exists $a \in A_i$ such that $a \notin B'_i$. Let $B''_i = B'_i \cup A_i \supset B'_i$. Clearly, B''_i is weakly truthful and contains a.

If $\mathit{filter}(i, (B''_i, B_{-i})) \cap A_i = \emptyset$, then B''_i and B'_i contain the same optimal offers w.r.t. $\leq^c$ (as none of them belongs to A_i), therefore $pw(i, (B'_i, B_{-i})) = pw(i, (B''_i, B_{-i}))$. This proves cond. 1 of APP.

Next suppose that $\mathit{filter}(i, (B'', B_{-i})) \cap A_i \neq \emptyset$. Since $B''_i \supset B'_i$, it follows that $\mathit{filter}(i, (B'', B_{-i})) \supseteq \mathit{filter}(i, (B', B_{-i}))$, by definition. Therefore B''_i can only add better optimal offers (w.r.t. $\leq^c$), or leave them unchanged in the worst case. Consequently, B''_i may cause i to enter the set of candidate winners and/or reduce the number of competitor candidate winners, so $pw(i, (B'', B_{-i})) \geq pw(i, (B', B_{-i}))$, which proves cond. 2 of APP.

Finally, cond. 3 follows immediately from the inclusion $\mathit{filter}(i, (B'', B_{-i})) \supseteq \mathit{filter}(i, (B', B_{-i}))$ (cf. the above paragraph). Then, the APP implies that $B'_i \preceq_i B''_i$, and we are only left to show that this preference is strict.

Consider the following choice of $\leq^c$ and B_{-i}. The auctioneer's preference relation $\leq^c$ is the least partial order extending the "obvious" preference relation $\leq$, such that if $b \not\leq a$ and $a \not\leq b$, then $b <^c a$. Note that for all SLA Templates $b \neq a$, either $a <^c b$ or $b <^c a$, and that $a <^c b$ iff $a < b$. Moreover, suppose that for all bidders $j \neq i$, $B_j = \{a\}$. It follows from the definitions and the properties of $<^c$ that $\mathit{filter}(i, (B'_i, B_{-i})) = \{b \in B'_i \mid a < b\}$ and $\mathit{filter}(i, (B''_i, B_{-i})) = \{b \in B''_i \mid a \leq b\}$. The former equality and (2) imply that for all $b \in \mathit{filter}(i, (B'_i, B_{-i}))$, $b <_i a$ holds. The latter equality implies that $a \in \mathit{filter}(i, (B''_i, B_{-i}))$. It follows that in some case the disequality in cond. 3 is strict, so $B'_i \prec_i B''_i$. ■

The above theorem implies a nice guarantee on auction outcomes, that usually is implied by truthfulness. Note that if bidders are rational (and hence weakly truthful, according to Theorem 4.1), then for all providers $i \neq j$, for all $b \in \mathit{filter}(i, \mathbf{B})$, and for all $a \in A_j$, we have $b \not<^c a$. Then, from a customer's viewpoint, the contract b_w selected by the winner w is never worse than any contract that the other bidders $j \neq w$ would offer *if they were truthful* (weak optimality):

$$\forall a \in \bigcup_{j \neq w} A_j \,,\; b_w \not<^c a \,. \tag{3}$$

Overbidding can only improve the outcome, from the customer's viewpoint, e.g. by removing some of the less desirable contracts from the filters, or even forcing w to stipulate a non-profitable contract ($b_w \notin A_w$) that is strictly better than some of the $<^c$-optimal contracts in A_w.

Our auction mechanism enjoys also a further classical, desirable property: Whenever the customer's request matches at least one service (formally, $A^c \cap \bigcup_i A_i \neq \emptyset$) and the winner w does not overbid, then the outcome of the auction is always a contract b_w that is profitable for w (i.e. $b_w \in A_w$); in other words, providers always have a concrete interest in participating to the auction.

Example 4.1: Let us consider three ASPs participating in a public e-auction. ASP_1 offers SLA Templates t_1, t_3, t_9, t_{10}, ASP_2 SLA Templates t_2, t_4, t_8, and ASP_3 SLA Templates t_5, t_6, t_7, t_8. Based on the ordering in Figure 3, ASP_1 is the winner of the e-auction. Following our approach, we note that ASP_1 can provide its final SLA contract choosing between bids t_1, t_9, and t_{10}. Bid t_3 cannot be chosen because it is weaker that bid t_2 from ASP_2. Assuming that ASP_1 also offers t_8, the latter could be used as the SLA contract, since no other ASPs than ASP_1 provided bid t_9.

V. Performance Evaluation

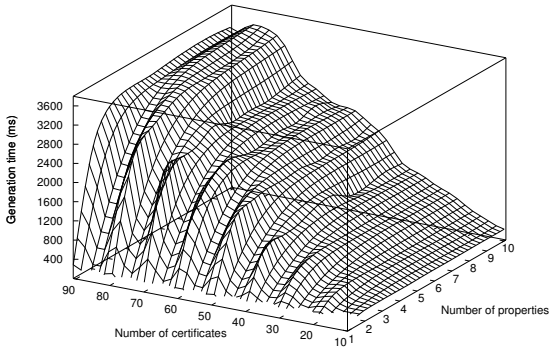
We implemented a Java-based prototype of our e-auction approach for the computation of the partial order of bids (Section III) and for template filtering (Section IV). We then evaluated the prototype performance by measuring its execution time, varying the number of bids from 10 to 90 (with a step of 10) and the number of non-functional properties in SLA Preferences and SLA Templates from 1 to 10. These ranges have been selected to minimize the impact of perturbations (e.g., due to memory swapping effect of the JVM) on the experimental performance evaluation. We further adopted a fixed abstract property prioritization for all the experiments to reduce biased results due to particular configurations of priorities and considered hierarchies $\mathcal{H}_{cat}$ with depth and width equal to 10. To make the performance evaluation as generic as possible, we also developed a SLA Template generator, which randomly generates SLA Templates on the basis of properties in the selected SLA Preference.

We considered two scenarios for performance evaluation as follows: *i)* all bids are generated randomly with negligible probability of having two bids with the same set of properties, *ii)* every bid has exactly one replica (e.g., in the case of 10 bids, we generated 5 different bids and replicated them to produce 10 bids). The first scenario is an approximation of a real e-auction environment, where the probability of having exactly the same bid from different ASPs is usually very low. The second scenario is an ad hoc scenario designed to exercise our matching and ranking process in a worst case scenario in terms of required computational resources. In the latter case, in fact, our algorithm produces n sets of two templates, with n equal to half of the number of templates under evaluation, and tries to order them by evaluating all properties in the templates.

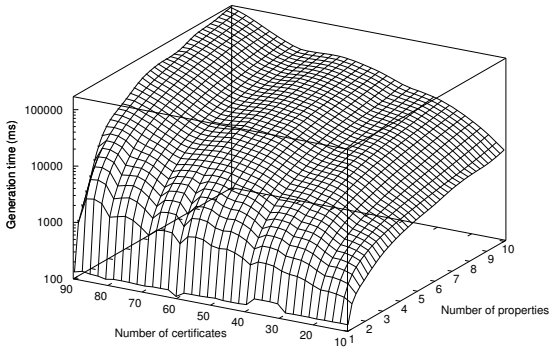
Experiments have been run on a *Mac Powerbook* equipped with one Intel i5 2.53GHz, 8GB of RAM, one 500GB disk, and Mac Os X 10.6.8 operating system. First, we measured the execution time for the computation of the partial order of bids. For every scenario, we repeated the tests ten times and measured the mean execution time over the ten executions. Figure 4(a) shows the performance of our prototype in scenario *i)*, where all executions require less than $3.49s$. Figure 4(b) instead shows, in logarithmic scale, the performance of our prototype in scenario *ii)*. In the worst case (90 bids and 10 properties), it returns the partial order to the auctioneer in $165.09s$. We note that in both scenarios the matching phase has negligible execution times (less than $27.5ms$ in the worst case of 90 bids with 10 properties). The difference in time between the two scenarios is mainly due to the fact that in the first case, given the random nature of the templates and the low probability of having replicated templates, a complete partial order is generated before all ordering processes have been executed. As an example, $\approx$3 ordering steps are required on average when 10 properties are used considering 10 executions over 90 bids. In the second step, instead, the presence of replicated bids force the algorithm to check all properties.

We then measured the execution time of the e-auction winner selection (template filtering in Section IV). Our experiments showed that the computational cost of this execution is marginal, requesting a maximum of 5ms when 90 templates are filtered to select the e-auction winner.

In summary, our experiments showed that the overhead given by our prototype is acceptable for an e-auction process, even in the worst case. We note that the number of bids and of non-functional properties in the SLA Preference we tested is very challenging, since e-auctions involving 90 offers and requirements on 10 properties are not usual. We also note that the performance of our e-auction service procurement prototype is comparable with recent query-based service procurement approaches based on similar matching and comparison steps, as for instance, the one in [11] on non-functional certificates. Neglecting the setup time in both



(a) Random bids



(b) Bids with 50% replica (logarithmic scale)

Figure 4. Performance evaluation: Ranking generation time

scenarios (i.e., certificate generation and e-auction setup), the main difference resides in the execution of additional filtering for the selection of the e-auction winner, which we showed to be marginal with respect to the overhead given by the computation of the partial order of bids.

VI. Related Work

Our e-auction-based service procurement shares a common ground with research in the area of non-functional service discovery and selection [12]. Grag et al. [13] consider generic non-functional properties and propose an Analytic Hierarchy Process-based framework. The proposed framework measures the quality of cloud services using a ranking approach based on QoS client's requirements and a service measurement index. Similarly to our selection approach, the solution in [11] proposes a two-step matching and ranking mechanism for service certification. In this paper, we consider a larger set of non-functional security properties and adopt SLA-based provisioning using a generalization of procurement e-auctions. Sakr and Liu [14] discuss the challenges and the importance of SLA-based provisioning, considering the scenario of cloud-hosted databases, and

propose a SLA-based management framework for database tier. The adoption of auctions in the cloud is at the basis of Marinescu et al.'s paper [15]. They adopt combinatorial auctions, where bids are bundles of items (e.g., combinations of CPU cycles, main memory, secondary storage), thus permitting a better control of QoS and security in the cloud architecture. Differently from our approach, they do not allow direct bidding on specific non-functional properties, focusing more on self-organization of the cloud architecture. Game-theoretic techniques for total preference orderings are also used in [16], to model SaaS providers' competition for the VMs of an IaaS provider. They do not provide any truthfulness guarantees nor similar quality properties. The inherent complexity of selecting an optimal set of services when preferences are determined by a real-valued utility function (such as cost) has been studied in [17]. Previous auctions for arbitrary, partial preference relations enjoy truthfulness under the assumption that the risks related to overbidding are high enough to prevent it completely [5] or under the implicit assumption that overbidding may not occur [4]. Here we followed a different approach. By not publishing the customer's preferences, and exploiting some natural relationships between the preferences of customers and bidders, we obtain contract quality guarantees (cf. (3)) without relying on any assumption about overbidding.

VII. Conclusions

Cloud-based service deployment and distribution are radically changing and playing a crucial role in today IT systems. However, the lack of appropriate solutions for truthful selection of services on the basis of their properties is increasingly perceived as a limitation, especially in those sectors having strict functional and non-functional requirements for software procurement. The approach presented in this paper aims at providing a solution that models the multi-cloud provisioning and selection scenarios as a generalization of second price procurement e-auctions. Our approach is based on a matching and ranking process, produces a partial order of bids provided by service providers in the e-auction according to the auctioneer preferences, and can be easily integrated in existing cloud infrastructures. Our approach guarantees at least the same benefits as truthfulness on the e-auction outcome, and its time performance is comparable to query-based selection engines.

Acknowledgments

This work was partly supported by the Italian MIUR project SecurityHorizons (c.n. 2010XSEMLC) and by the EU-funded project CUMULUS (contract n. FP7-318580).

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," in *Tech. Rep. UCB/EECS-2009-28*, EECS Department, U.C. Berkeley, February 2009.

[2] P. Bosc, E. Damiani, and M. Fugini, "Fuzzy service selection in a distributed object-oriented environment," *IEEE TFS*, vol. 9, no. 5, pp. 682–698, 2001.

[3] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[4] P. Bonatti, M. Faella, C. Galdi, and L. Sauro, "Towards a mechanism for incentivating privacy," in *Proc. of ESORICS 2011*, Leuven, Belgium, September 2011.

[5] ——, "Auctions for partial heterogeneous preferences," in *Proc. of MFCS 2013*, Klosterneuburg, Austria, August 2013.

[6] M. Galster and E. Bucherer, "A taxonomy for identifying and specifying non-functional requirements in service-oriented development," in *Proc. of IEEE SERVICES 2008*, Honolulu, HI, USA, July 2008.

[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE TDSC*, vol. 1, no. 1, pp. 11–33, 2004.

[8] S. Ran, "A model for web services discovery with QoS," *ACM SIGecom Exchanges*, vol. 4, no. 1, pp. 1–10, 2003.

[9] E. Casalicchio and L. Silvestri, "Mechanisms for SLA provisioning in cloud-based service providers," *Computer Networks*, vol. 57, no. 3, pp. 795–810, 2013.

[10] C. Ardagna, E. Damiani, K. Sagbo, and F. Frati, "Zero-knowledge evaluation of service performance based on simulation," in *Proc. of IEEE HASE 2014*, Miami, FL, USA, January 2014, short paper.

[11] M. Anisetti, C. Ardagna, E. Damiani, and J. Maggesi, "Security certification-aware service discovery and selection," in *Proc. of IEEE SOCA 2012*, Taipei, Taiwan, December 2012.

[12] H. Yu and S. Reiff-Marganiec, "Non-functional property based service selection: A survey and classification of approaches," in *Proc. of NFPSLAM-SOC 2008*, Dublin, Ireland, November 2008.

[13] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1012–1023, June 2013.

[14] S. Sakr and A. Liu, "SLA-based and consumer-centric dynamic provisioning for cloud databases," in *Proc. of IEEE CLOUD 2012*, Honolulu, HI, Hawaii, June 2012.

[15] D. C. Marinescu, A. Paya, J. P. Morrison, and P. D. Healy, "An auction-driven self-organizing cloud delivery model." *CoRR*, vol. abs/1312.2998, 2013.

[16] D. Ardagna, B. Panicucci, and M. Passacantando, "A game theoretic formulation of the service provisioning problem in cloud systems," in *Proc. of WWW 2011*, Hyderabad, India, March-April 2011.

[17] P. Bonatti and P. Festa, "On optimal service selection," in *Proc. of WWW 2005*, Chiba, Japan, May 2005.