Reliable QoS Monitoring Based on Client Feedback

Radu Jurca
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Lab
Lausanne, Switzerland
radu.jurca@epfl.ch

Walter Binder
University of Lugano
Faculty of Informatics
Lugano, Switzerland
walter.binder@unisi.ch

Boi Faltings
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Lab
Lausanne, Switzerland
boi.faltings@epfl.ch

ABSTRACT

Service-level agreements (SLAs) establish a contract between service providers and clients concerning Quality of Service (QoS) parameters. Without proper penalties, service providers have strong incentives to deviate from the advertised QoS, causing losses to the clients. Reliable QoS monitoring (and proper penalties computed on the basis of delivered QoS) are therefore essential for the trustworthiness of a service-oriented environment. In this paper, we present a novel QoS monitoring mechanism based on quality ratings from the clients. A reputation mechanism collects the ratings and computes the actual quality delivered to the clients. The mechanism provides incentives for the clients to report honestly, and pays special attention to minimizing cost and overhead.[1]

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; K.4.4 [**Computers and Society**]: Electronic Commerce

General Terms

Algorithms, Economics, Measurement, Performance, Security

Keywords

Service-oriented computing, service-level agreement, quality-of-service, reputation mechanism, incentive compatibility

1. INTRODUCTION

Service-oriented computing enables the construction of distributed applications by integrating services that are available over the web [21]. The building blocks of such applications are web services[2] that are accessed using standard protocols.

In this paper we assume a service market where services are accessed according to service-level agreements (SLAs). SLAs are advertised in directories by service providers. A SLA identifies the service provider and includes information concerning service functionality and grounding, which may be specified in a formalism such as WSDL [26], OWL-S [19], or WSMO [28]. Moreover, a SLA specifies the conditions of service delivery, such as the price for service invocation as well as Quality of Service (QoS) parameters (e.g., maximum response time). Languages such as Web Service Level Agreements (WSLA[3]) [5] or WS-Agreement [2] may be used to specify such non-functional properties. Service directories offer matchmaking functionality allowing clients to discover SLAs that fit their requirements.

Essential for the functioning of such a service market is the credibility of SLAs. Unreliable SLA advertisements decrease the overall welfare of the market, since clients do not have accurate information to plan their business. As clients are usually required to pay for a SLA before receiving the requested service, providers have an opportunity to cheat. They may provide lower QoS than advertised, and thus save costs. It is therefore necessary to create incentives for service providers to respect their advertised SLAs by stating penalties that must be paid when the delivered QoS is less than promised.

In order to enforce such penalty payments, the market has to provide effective mechanisms to monitor QoS in an objective and reliable way. There is a large body of research addressing infrastructural facilities for SLA monitoring [24, 12, 5, 3]. Existing solutions rely on one of the following three techniques:

- a trusted *monitor* intercepts the messages exchanged between the client and the provider and outputs an estimate of the delivered QoS.
- monitoring code runs on the provider side, as part of the service middleware. The monitoring layer intercepts the messages addressed to/originating from the provider, and estimates the delivered QoS.
- a trusted party periodically probes the service and outputs performance metrics.

The problem with the first technique is scalability. When the monitor intercepts all service invocations, it acts as a central proxy and soon becomes a performance bottleneck. Bottlenecks may of course be avoided by only monitoring a

[1]The work presented in this paper was supported by the Swiss National Funding Agency OFES as part of the European project KnowledgeWeb (FP6-507482).
[2]We use the terms *web service* and *service* interchangeably.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

[3]http://www.research.ibm.com/wsla/

sample of the service invocations, but the monitoring will be less precise.

The problem with the second techniques is trustworthiness. The providers have obvious strategic incentives to modify the monitoring results. Unless strongly secured (which comes at a non-negligible cost), the monitoring code may be tampered with, and rendered unreliable.

The third technique is expensive and probably inaccurate. Special clients must be set up only to probe and evaluate the service. They generate supplementary service requests which unnecessarily overload service providers. Moreover, trusted clients monitor only a small sample of the total number of requests, and therefore the output results are prone to noise and errors.

In this paper we introduce an alternative QoS monitoring mechanisms based on feedback provided by clients. In our solution, the clients are running the monitoring code, and periodically report feedback to a trusted center (referred to as the *reputation mechanism* or RM). The RM aggregates the reports and estimates the delivered QoS for each provider. In this way,

- the RM can get information about most transactions without actually being a bottleneck (there are no real-time constraints for reporting feedback, and the result of several interactions may be compressed in one feedback message);
- the monitoring process is as precise as possible (an immediate consequence of the first point);
- the provider cannot directly tamper with the monitoring process;

Accurate mechanisms must, however, address two problems. The first, is obtaining honest feedback reports. We rely on economic incentives rather than hard security measures, and thus make lying *uninteresting* rather than *impossible*. Honest reporting incentives are created through a payment mechanism where every client gets paid for submitting feedback an amount that depends on the collective set of feedback received by the RM in a certain time-window. We prove that truthful reporting maximizes the expected revenue (due to feedback payments) of a client, motivating an equilibrium where every client reports honestly.

The second problem is collusion. The payment mechanism makes *individual* honest reporting rational, however, several clients that coordinate on a lying strategy can still manipulate the monitoring results without suffering lower expected payments. We therefore modify the initial payments to also be robust against coalitions that are smaller than a certain threshold.

The paper is organized as follows. In Section 2 we describe the general assumptions behind our environment. Section 3 presents the interaction protocol and Section 4 describes some of the implementation details of a monitoring framework prototype. Section 5 presents a payment mechanism that the RM can use to make rational agents report the truth. The robustness against colluding reporters is addressed in Section 6, followed by an example in Section 7. Finally, Section 8 compares our results with related work.

2. SETTING AND ASSUMPTIONS

We consider an online market of services [21] where different clients interact with different service providers in a decentralized manner. There is no trusted authority or proxy intermediating the transactions between clients and providers, except that service discovery is facilitated by directories. Both clients and providers have digital identities based on public key infrastructure. The complete upper level interaction protocol is described in Section 3.

Services are characterized by binding SLAs specifying both functional and non-functional (quality) attributes [5, 2]. We divide time into equal periods and assume that the same SLA is shared by a large group of clients in any given period of time. The same service provider can have several customer groups, but all clients within the same group are treated equally (within the same period of time). The length of the time period is an application-dependent parameter, set to meet the two constraints above (i.e., large number of client requests per period, but the same SLA and service parameters).

The Quality of Service (QoS) is specified according to a common ontology such as [15] or [23]. We impose, however, several restrictions on the type of QoS descriptions that occur in the SLAs. First, we consider only *objective* quality attributes that take discrete values, and can be observed by clients for single service invocations. *ServiceIsAlive* or *InvocationFailure* are examples of such quality attributes: they are understood by all agents in the same way, can be measured for each interaction, and take boolean values. *ResponseTime* and *Bandwidth* are both objective and observable, but usually take continuous values. For most applications, however, clients are indifferent between values that fall within some range, and therefore, they can be discretized: e.g., *Bandwidth* $\in$ { *DialUp, DSL, T1*}. On the other hand, *Availability* or *Reliability* do not meet our restriction since they are not observable for single interactions.

Second, we assume that quality properties are specified in the SLA as probability distributions over possible values for each of the quality attributes. *Availability* can therefore be indirectly expressed as a probability distribution over the boolean values of the quality attribute *ServiceIsAlive*. Such descriptions can be regarded as simple extensions to the formalism used in [15, 23, 20], where quality attributes are characterized by min, max and/or typical values.

Finally, we assume that the values of different quality attributes are independent, with the only exception that certain values of certain quality attributes render the observation of other quality attributes impossible: e.g., if for the present invocation the *ServiceIsAlive* attribute has the value *FALSE*, the value of the *ResponseTime* attribute cannot be observed.

While simplified, we believe that our model is still general enough to be of practical use in many domains. Furthermore, the assumption that quality attributes are independent (with the exception mentioned in the previous paragraph) can be relaxed without any theoretical difficulties. Nonetheless, the appropriate notation that would allow us to formally explain the effect of correlated quality attributes on the reporting incentives is cumbersome, and dependent on the particular application. At the end of Section 5 we provide an informal discussion of how to extend the mechanisms presented in this paper for the more general model including correlations.

Formally, let $Q = \{q_1, q_2, \ldots, q_n\}$ be the set of all quality attributes defined by our ontology, and let V_i be the domain of values of the quality attribute q_i. We assume there is a

strict total order over the elements of V_i, such that $v_j < v_{j+1}$ (the value v_{j+1} is preferred by all clients to value v_j) for all j. Generally, the dependence between quality attributes is expressed through a (linear) correlation factor between the values of those attributes [15]. With our simplifying assumptions, however, this dependence can be expressed as a relation:

$$\mathcal{R} = \{(q_i, v_i, q_j) | q_i, q_j \in Q, v_i \in V_i\};$$

specifying all tuples (q_i, v_i, q_j) such that when the quality attribute q_i takes the value $v_i \in V_i$, the quality attribute q_j cannot be observed. For example, the relation $\mathcal{R}$ may contain the tuple (*ServiceIsAlive, FALSE, ResponseTime*) since the response time of a service that is not alive cannot be observed.

A *description* of the quality attribute q_i is a cumulative probability distribution $\pi_i : V_i \to (0, 1)$ over all possible values of the attribute. For example, a description of the quality attribute *ResponseTime* could be the following: "*the response time is: less than 0.1s with probability* 30%*, less than 0.5s with probability 70%, and less than 1s with probability* 100%".

A *quality advertisement*, as published by a SLA, describes a subset $\bar{Q} \subseteq Q$ of quality attributes.

Service providers are rational, and they can advertise a false QoS. To overcome this problem, SLAs can be extended with a clause that punishes providers for not keeping their promises. The SLA defines the penalties that must be paid by the provider to the client if the delivered QoS is less than advertised. [8] shows that appropriately scaled penalties that depend on the difference between the delivered and the advertised QoS make it rational for all providers to advertise the QoS honestly.

Our monitoring mechanism relies on the clients to provide the information required to estimate the delivered QoS. After every interaction, the client observes a value for some (or possibly all) of the quality attributes specified in the SLA. A *quality observation* is a vector containing a value for each of the quality attributes specified in the SLA: i.e., $o = (v_i)$, where $v_i \in V_i \cup \{null\}$ for all $q_i \in \bar{Q}$. Since not all combinations of values can occur simultaneously (because of the constraints defined by the relation $\mathcal{R}$), the quality attribute q_j will have the value $v_j = null$ if and only if some other quality attribute q_i has the value v_i, and $(q_i, v_i, q_j) \in \mathcal{R}$.

A trusted RM is responsible for gathering and aggregating the feedback from the clients. The feedback is used to compute the delivered QoS and to update (in an application-dependent manner) the reputation information about the service provider. The RM publishes periodically, at the end of every period of time, the monitored value for the QoS. When the monitored QoS is less than advertised in the SLA, all clients that received the service in the last period are entitled to penalties paid by the corresponding providers. The providers that do not pay their penalties are excluded from the market: e.g., are listed on *black lists* and will be avoided by future clients.

The feedback messages submitted by the clients consist of a set of quality reports about the interactions between the client and service providers. One message can thus compress information about the several transactions with several service providers. We assume that quality observations can be derived automatically from the messages exchanged between the client and the provider. To facilitate the reporting, the RM makes available the monitoring and reporting code that allows the clients to automatically submit feedback.

Although feedback is by default reported honestly, clients can tamper with the reporting code when they increase their utility by doing so. Let $\Delta > 0$ be an upper bound on the utility increase an agent can obtain by lying, as, for example,

- falsely reporting low quality decreases the reputation of the provider, who may be forced, in the future, to decrease the price of service;
- the decreases in reputation due to a false report may also drive away other clients, leaving the service provider more available to the requests of the lying agent;
- falsely reporting high quality could attract rewards or preferential treatment from the provider.

Tampering with the reporting code is costly, and we denote this cost by C. The same modified code can be used repeatedly or shared by several clients, and therefore, the marginal cost of one false report is often smaller than Δ. The potential advantage a client can obtain by lying motivates the need for measures to ensure honesty.

Opposed to traditional techniques, our approach is to make lying uninteresting, rather than impossible. We use a minimum of cryptographic tools, and propose a payment mechanism that rewards honesty. The RM will pay something for every submitted feedback, and the payments will be scaled such that, in expectation, the reward from telling the truth is better than the reward when lying by at least Δ. This property guarantees that no agent (or small coalition of agents) has the incentive to tamper with the reporting code.

However, before presenting in more detail the incentives that drive clients to report honestly, Section 3 presents the interaction protocol and Section 4 gives more implementation details of the QoS monitoring framework.

3. INTERACTION PROTOCOL

The participants in our environment are the following: *service providers* advertise SLAs and offer the corresponding services; *clients* choose SLAs and invoke the respective services; *service directories* facilitate the matching between clients and providers; *RMs* collect and aggregate feedback from the clients; a *bank* handles payments. The RMs and the bank are trusted parties. A RM can be integrated into a service directory in order to enable efficient, reputation-aware SLA selection. In this case, the service directory integrating a RM is assumed to be trusted.

Figure 1 illustrates the interactions between the aforementioned participants:

1. Providers advertise SLAs to a service directory (1a). Each SLA uniquely identifies the service provider and the service functionality, for example by referring to a WSDL service description, and defines the price and QoS for service invocation. The service directory assigns a suitable RM for each SLA advertisement, which shall be used for feedback reporting. The instantiation of a RM for a new SLA (1b) requires solving a linear optimization problem, which will be discussed in Section 5. Advertised SLAs remain valid for a period of time specified by the provider. After expiration, they are removed from the directory. Service directories may support leases, allowing service providers to refresh

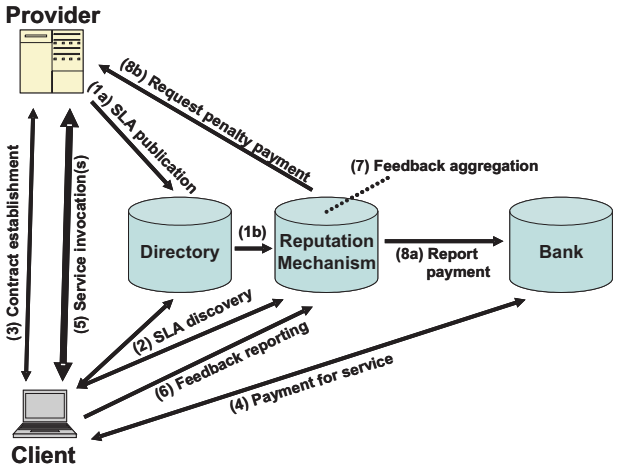

Figure 1: Interaction protocol involving a RM.

SLA advertisements. Each SLA receives a unique *SLA-ID*, computed as a secure hashcode of the SLA.

2. Clients search for advertised SLAs according to functional and non-functional criteria, as well as according to reputation information. To this end, clients access a directory and a RM. If the RM is integrated within the directory, reputation-based filtering constraints can be directly included in the directory query. Clients may inspect reputation information specific to a SLA, or aggregated reputation information for a service provider.

3. The client and the chosen provider establish a contract for a given SLA, for a given period of time. The client sends a request message to the service provider, including *Client-ID*, *SLA-ID*, and the number of requested service invocations, *Nr-Invoc*. The service provider may reject the request, if it (temporarily) cannot meet the conditions of the SLA. The response message sent by the service provider is a non-forgeable service invocation capability (SIC), valid for *Nr-Invoc* service invocations according to the conditions advertised in the SLA *SLA-ID*. The SIC will also be used by the client to report feedback.

4. The client pays for the agreed number of service invocations (i.e., *Nr-Invoc* times the price stated within the SLA). The payment message includes the SIC, and the bank returns the signed SIC in order to certify successful payment.

5. The client requests the service, and the provider responds. For each service invocation, the client has to provide a valid SIC signed by the bank. Hence, the service provider can easily determine that the client has payed for the SLA. The service provider keeps track of the number of service invocations for each valid SIC in order to ensure that this number does not exceed the contracted *Nr-Invoc* value. The client monitors the QoS parameters to be reported to the RM.

6. The client sends feedback to the RM. The feedback contains the SIC signed by the bank, and a timestamped series of quality reports. For each SIC, the client may send between 1 and *Nr-Invoc* reports. The quality reports need not necessarily be aggregated within a single message. I.e., for the same SIC, the client may send several messages with a varying number of quality reports. The RM does not verify whether a service was actually invoked by the client, but it ensures that the client paid for the invocation. I.e., the RM rejects reports if the SIC has not been signed by the bank.

7. The RM aggregates received feedback at the end of each time period. From all valid quality reports about a SLA, the RM estimates the actually delivered QoS by computing the distribution of values (i.e., histogram) for every quality attribute described by the SLA. Feedback can also be used to update the reputation of the service provider.

8. The RM pays valid reports as described in Section 5 (8a). Finally, the RM publishes the monitored QoS value for the current period and notifies the providers about the penalties they must pay (8b). Service providers who do not pay the agreed penalties may be put on a *black list* by the RM and consequently will be avoided by clients upon service selection.

4. IMPLEMENTATION OF A PROTOTYPE

To validate the model discussed in the previous sections, we implemented a prototype of the QoS monitoring framework as a light-weight add-on on top of existing web-service middleware (Axis[4]). The framework exposes three types of components: directory services, reputation mechanisms, and banks. It also uses external certification authorities in order to setup a public key infrastructure (PKI). The users of the framework (i.e., the clients and the service providers) are provided with appropriate libraries in order to facilitate the deployment of applications.

As a general principle, all components expose two kind of interfaces:

- a web service exposing the functionality available to the users (providers and clients) of the framework. We will refer to this web service as the *public* web service (respectively the *public* interface).
- a web service exposing the functionality available to the other components within the framework. For security reasons, providers and clients do not have access to this web service. By abusing the terminology we will refer to this web service as the *private* web service (respectively the *private* interface).

The certification authority (CA) must provide the standard functionality associated to this role: creation and signing of digital $X.509$ certificates, validation of certificates, and revocation of expired or compromised certificates. For testing purposes we implemented a demo CA in our framework, however, any CA with a web service interface may be used. All parties (clients, providers, as well as directory services, reputation mechanisms and banks) are required to have valid identity certificates; these will be used to sign, encrypt, and authenticate exchanged messages. In our current version, we assume that CAs enforce unique identities and unique names.

The directory service is implemented as a wrapper around one or several UDDI and WSLA repositories. The public interface of the directory allows service providers to register, modify and delete service descriptions and service level agreements. Service registrations are requests by providing standard WSDL documents, signed by the provider. The directory checks the validity of the signature, and forwards the request to the UDDI repository (we used JUDDI[5] as

[4]http://ws.apache.org/axis/
[5]http://ws.apache.org/juddi/

the implementation of UDDI). The business key returned by the UDDI repository is returned to the provider, but is also stored by the directory next to the identity of the provider. Any subsequent modifications to existing service descriptions are first validated by the directory in order to avoid malicious corruption of WSDL documents.

Service providers may announce several SLAs for the same service. The registration of one or several SLAs is made by providing one, respectively several WSLA documents, describing the non-functional characteristics of the service. The directory first checks the validity of the business key against the identity of the provider, and then forwards the request to a proprietary WSLA repository[6]. The WSLA document describes the quality attributes of the service, by providing a cumulative distribution function on the values of each attribute. The quality attributes and possible values are described in an ontology.

Clients can search the directory for services that fulfill functional and non-functional requirements. Non-functional requirements are specified as a list of constraints that must be simultaneously met. Every constraint specifies a tuple (q_i, v_j, p_k), meaning that the client expects for the quality attribute q_i a value higher than v_j with probability greater than p_k. Efficient queries of the WSLA repository can be implemented by indexing the WSLA documents according to all possible tuples (q_i, v_j).

The private interface of the directory is used by the Bank to signal the service providers that pay (or do not pay) the required penalties. Service providers that refuse to pay the penalties are eventually placed on a black list, and are excluded from the result set returned to the clients.

Among the modules provided by our framework, the bank is the simplest one. The public interface of the bank includes the traditional operations (i.e., account creation, deposits, balance checks and withdrawals) as well as two functions required to support the interaction protocol in Figure 1. The first, `paySIC(SIC, signatureOfClient)` is used by clients to pay for a service invocation capability (SIC) in step 4 of the interaction protocol. The bank signs the SIC as a proof of payment, and returns it the client. The second, `payPenalty(bill, signatureOfProvider)` is used by providers to pay the penalties resulting from delivering lower than advertised QoS (step 8). The bills are created periodically by the reputation mechanism, and reflect the difference between the advertised and delivered quality levels. Providers can instruct the bank to automatically pay the penalty bills.

The private interface of the bank allows the reputation mechanism to announce the penalties that should be paid by a provider for not respecting the terms of the SLA. In response to such announcements the bank notifies the provider about the pending payment, or automatically pays the penalty if instructed so by the service provider.

Clients submit feedback reports by using the public interface of the reputation mechanism. One message may contain a set of reports, made up of:

- quality observations (as defined in Section 2) for one or several SLAs
- the corresponding SICs, signed by the bank
- the signature of the client.

The RM checks the validity of the client's signature, and verifies the signature of the bank on the SIC. All reports about a SLA beyond the number specified in the SIC are discarded.

The private interface of the reputation mechanism is used by the directory in order to query the reputation of certain service providers.

All components of the framework also include code for house-keeping operations like maintenance of databases, purging expired records, revoking expired certificates, etc. This code can run either as a separate demon process when the middleware allows it, or as part of the calls to the public or private web services.

5. INCENTIVES FOR TRUTHFUL REPORTING

An essential component of our mechanism is a payment scheme that rewards clients for honestly reporting feedback. Such payments can be constructed by comparing every report with some other report (called the *reference report*) submitted by a different client about the same SLA. As the two reports refer to the same service, there is a link between them; this link will be exploited such that whenever the reference report is true, it also becomes in the reporter's best interest to report the truth. Honest reporting thus becomes a Nash equilibrium in our environment [17].

The simplest payment rule pays a report only if it matches (i.e., has the same value as) the reference report. For example, a negative report about the SLA that describes only the attribute *ServiceIsAlive* is paid only if the reference report is negative as well. However, the payments depend on the actual value of report, and a negative report is paid differently from a positive report.

The reason why such payment rules encourage truthful reporting can be explained by the subtle changes in beliefs triggered by the private experience of a client with a given service provider. Although clients know that the service provider has all the incentives to deliver the promised QoS, they also realize that the delivered QoS will only *in expectation* equal the advertised one. Environmental noise and other unpredictable events will perturb the delivered QoS, making it higher in some rounds, and smaller in others. This is why the current experience of a client also conveys information about the immediately future interactions of other clients with the same provider.

It is therefore an acknowledged empirical fact [22] (also in full accordance with Bayesian theory) that an agent's posterior belief about the observation of another client (that receives the service in the same conditions) depends on the private experience of the agent.

For example, a client that has just had a negative experience believes that the clients in the same round will probably experience similar problems. Thus, she expects that the reference report used to compute her payment from the RM will correspond to a QoS that is slightly lower than advertised. On the contrary, a satisfied client is more likely to believe that other clients will be satisfied as well; therefore, she expects a reference report corresponding to slightly higher QoS than advertised.

[6]Our current implementation uses a MySQL database to implement the WSDL repository, with appropriate indexes to facilitate the search of services with the desired quality levels

The payments are designed such that a negative report maximizes the expected return only when clients expect a negative reference report with probability higher than advertised. And vice-versa for a positive report. Given that the reference report is true, the client maximizes her returns by reporting honestly, which makes truth-telling a Nash equilibrium. This means that no agent can gain an advantage by deviating from the protocol. Miller et al. [17] present a game theoretic analysis of such reporting scenarios and show that it is always possible to choose payments that make truth-telling optimal.

Concretely, our RM computes the payment made to every client in the following way. At the end of every period, all quality reports about the same SLA are grouped in a single set. Remember that each report corresponds to a quality observation, and therefore consists of an array of values, each corresponding to a quality attribute advertised by the provider.

For each report, $r = (v_i^r)$, the RM randomly takes a reference report, $rr = (v_i^{rr})$, coming from a different client. Every pair of matching non-null values for the attribute q_i (i.e., $v_i^r = v_i^{rr} \neq null$) contributes with $\tau_i(v_i^r)$ to the payment for the report r.

If (r_j^a) are all the reports submitted by client a, and (rr_j^a) are the corresponding reference reports, the total payment received by a is:

$$Pay(a) = \sum_j Pay(r_j^a, rr_j^a); \quad Pay(r, rr) = \sum_i \tau_i(v_i^r, v_i^{rr});$$

$$\text{where: } \tau_i(v_i^r, v_i^{rr}) = \begin{cases} 0 & \text{if } v_i^r \neq v_i^{rr} \text{ or } v_i^r = null \\ \tau_i(v_i^r) & \text{if } v_i^r = v_i^{rr} \end{cases}$$

The payment mechanism is fully specified by announcing the amounts $\tau_i(v_i)$, paid for a report matching the reference report on the value v_i of the quality attribute q_i.

We compute the payment mechanism through *automated mechanism design* [4]. Instead of a closed form specification, we define the mechanism through a set of constraints that act on the decision variables (i.e., the payments $\tau(\cdot,\cdot)$ in our case). By adding an objective function, we get an optimization problem that solves for the best possible payment mechanism in a given context.

The optimal payment mechanism minimizes the total cost of the RM, while guaranteeing that honesty is better than lying by at least the desired margin. The cost of the RM will depend on the SLA, so the payment mechanism must be instantiated for every SLA.

The expected cost for an honest report equals the weighted sum of all amounts $\tau_i(v_j)$. The probability that payment $\tau_i(v_j)$ is made equals the probability that both the report and the reference report have the value v_j for the quality attribute q_i. Since each probability equals $\pi_i(v_j)$ and the two events are assumed independent, we have:

$$E[Cost] = \sum_{q_i \in \bar{Q}} \sum_{v_j \in V_i} \tau_i(v_j)\pi_i(v_j)^2; \tag{1}$$

To compute the expected revenue obtained by a client when lying or telling the truth, we must first describe the belief of a client regarding the reference report chosen by the reputation mechanism. Given the real quality observation $o = (v_i)$, we assume that the belief regarding the reference report change slightly in the direction of o. If $\pi_i(v_j)$ is the advertised probability that the attribute q_i takes the value v_j, the belief of the client assigns:

- at least the probability $\pi_i(v_i) + \big(1 - \pi_i(v_i)\big)\bar{\gamma}$ to the event that the reference report also has the value v_i for the attribute q_i, and,
- at most the probability $\pi_i(v_j)(1-\underline{\gamma})$ to the event that the reference report has some other value $v_j \neq v_i$ for the attribute q_i.

If $v_i = null$ (no observation was possible for the attribute q_i), we assume that the beliefs regarding the reference report remain unchanged. Both $\bar{\gamma}$ and $\underline{\gamma}$ take values between 0 and 1, and depend on the specific applications.

Honest reporting can be guaranteed when:

- for any quality attribute q_i, truthfully reporting maximizes the expected payment by at least δ:

$$\begin{aligned} \big[\pi_i(v_i) + \big(1 - \pi_i(v_i)\big)\bar{\gamma}\big]\tau_i(v_i) > \\ \Big[\pi_i(v_j)(1-\underline{\gamma})\Big]\tau_i(v_j) + \delta; \end{aligned} \tag{2}$$

  for all $q_i \in \bar{Q}$ and all $v_i \neq v_j \in V_i$,

- dependencies between quality attributes do not break the honest reporting incentives (i.e., since matching *null* values do not contribute to the payment, the payments for the values that cause the *null* reports must be large enough):

$$\begin{aligned} [\pi_i(v_i) + (1 - \pi_i(v_i))\bar{\gamma}]\,\tau_i(v_i) > \\ \Big[\pi_i(v_j)(1-\underline{\gamma})\Big]\,\tau_i(v_j) + \pi_k(v_l)\tau_k(v_l) + \delta; \end{aligned} \tag{3}$$

  for all $q_i, q_k \in \bar{Q}$, $v_i \neq v_j \in V_i$, $v_l \in V_k$ such that the value v_i of the attribute q_i makes impossible the observation of the attribute q_k: i.e., $(q_i, v_i, q_k) \in \mathcal{R}$.

The margin δ must offset the worst case incentive for lying. This value is very conservative and can be relaxed in real applications by considering that not all lies can simultaneously attract the worst case incentives.

The objective function in (1) together with the constraints defined by (2) and (3) define a linear optimization problem that accepts as a solution the cheapest incentive-compatible payment mechanism for a given SLA. The number of variables is equal to the overall number of values in all domains V_i, i.e., $\sum_{q_i \in \bar{Q}} card(V_i)$. The number of constraints is on the order of $\sum_{q_i \in \bar{Q}} card(V_i)^2$.

Extending our framework to include other types of dependencies or correlations between quality attributes does not pose theoretical challenges. Optimal payments that ensure truthful reporting can still be computed by extending the optimization problem with constraints like (3) that limit the gains of lying on pairs of values for correlated quality attributes. Intuitively, the additional constraints isolate independent *groups* of attributes, and guarantee truth-telling incentives. However, the notation that allows the definition of such payments is complicated, and outside the space limits set for this paper.

The payments naturally decrease with the margins, δ, required for truth-telling. Therefore, the expected cost of the RM can be decreased either by decreasing the benefits clients may obtain by manipulating their reports, or, by increasing the cost of tampering the reporting code. While the latter direction is outside the scope of this paper, the following two ideas can be used to address the former.

First, we can make sure that the penalties paid back by the provider do not give incentives to underrate the quality of the service. For that, we impose that the penalty paid to client a depends only on the QoS delivered to all other clients, except a. Given a large enough number of agents, the penalties paid by the provider are expected to be the same, but the feedback reported by a has no influence on the penalties paid to a.

A second interesting direction is to filter out the reports that are very far from the common distribution [9]. Intuitively, these reports are either erroneous, or intentionally try to introduce significant perturbations towards desired values.

6. COLLUSION AMONG CLIENTS

The payments defined in the previous section do not have truthful reporting as the unique equilibrium. Always reporting the same values is also an equilibrium strategy, since reports will surely match the corresponding reference reports. Moreover, it is easy to see that such *constant reporting strategies* yield higher payments than the truthful reporting. Fortunately, such coalitions on lying strategies can be rendered unprofitable when a fraction of reports are guaranteed to be honest.

We believe it is reasonable to rely on some fraction of truthful reports for several reasons. First, empirical studies show that a non-negligible fraction of users are altruists who always report the truth. Second, given that the framework already provides the default (honest reporting) code, some clients won't have the knowledge to temper with the reporting code even if they want to. Third, the reputation mechanism can probatively get honest reports by contracting specialized monitors to probe the service [8]. When the fraction of honest reports is large enough, individual clients cannot improve their payment by lying.

The idea behind fighting lying coalition is to make them unstable. We start from the assumption that at most $\lambda \in (0,1)$ percent of the clients can collude on a lying strategy. Then we compute a payment scheme that makes it individually better for a colluder to shift to the honest reporting strategy, knowing that $1-\delta$ percent of the reports are honest. Since the other coalition members cannot detect (and punish) deviators, all rational colluders will break the coalition and report honestly. The coalition is unstable in the sense that it is not profitable for coalition members to keep their commitment to the coalition.

Let us analyze the additional constraints on the optimization problem defining the payments. An honest reporter now expects that the reference report will be part of a coalition with probability at most λ. To make sure that the client still has the incentive to truthfully report the observed value v_i instead of the collusion value v_j, the constraint in (2) becomes:

$$\begin{aligned}(1-\lambda)\left[\pi_i(v_i)+(1-\pi_i(v_i))\bar{\gamma}\right]\tau_i(v_i) > \\ (1-\lambda)\left[\pi_i(v_k)(1-\underline{\gamma})\right]\tau_i(v_j)+\lambda\tau_i(v_k)+\delta;\end{aligned} \quad (4)$$

for all $q_i \in \bar{Q}$ and all $v_i \neq v_j \in V_i$. Similarly, the constraint (3) becomes:

$$\begin{aligned}(1-\lambda)\big[\pi_i(v_i)+(1-\pi_i(v_i))\bar{\gamma}\big]\tau_i(v_i) > \\ (1-\lambda)\Big[\big[\pi_i(v_j)(1-\underline{\gamma})\big]\tau_i(v_j) \\ +\pi_k(v_l)\tau_k(v_l)\Big]+\lambda\big(\pi_i(v_j)+\pi_k(v_l)\big)+\delta;\end{aligned} \quad (5)$$

for all quality attributes $q_i, q_k \in \bar{Q}$, values $v_i \neq v_j \in V_i$, $v_l \in V_k$, and tuples $(q_i, v_i, q_k) \in \mathcal{R}$.

The linear problem that minimizes (1) under the set of constraints (4) and (5) defines the incentive-compatible payments that are also λ-coalition proof.

7. EXAMPLE

We exemplify the mechanism described above with a simple weather service. The client submits a geographical location and the service returns the weather forecast for the next time interval.

The SLA advertises *availability* p_1 (i.e., the probability that a request is answered before a deadline t_d is p_1) and *correctness* p_2 (i.e., the probability of returning the correct information[7] is p_2). Formally, this SLA is expressed as the probability distribution $\pi_1 = \{p_1, 1-p_1\}$ for the quality attribute:

$$Q_1 = ResponseBeforeDeadline \in V_1 = \{0(false), 1(true)\};$$

and the probability distribution $\pi_2 = \{p_2, 1-p_2\}$ for the quality attribute:

$$Q_2 = InformationIsCorrect \in V_2 = \{0(false), 1(true)\};$$

Naturally, the relation $\mathcal{R}$ defining the dependency between quality attributes contains only the tuple $(Q_1, 0, Q_2)$: if no response is received, checking for correct information is meaningless.

A quality observation (and therefore a quality report) is a vector $o = (v_1, v_2)$ where $v_1 \in \{0,1\}$ and $v_2 \in \{0,1,null\}$. The payment scheme used by the RM is defined by the four positive amounts $\tau_1(1), \tau_1(0), \tau_2(1)$ and $\tau_2(0)$, paid when the non-null value of Q_1 or Q_2 matches the corresponding value of the reference report. The maximum benefit a client can obtain by misreporting one observation is $\Delta = 0.01$ (all values hereafter are normalized to the price of service, assumed 1), and the cost of tampering with the default monitoring code is $C = 10$. A client is assumed to generate at most $N = 1000$ service requests within the same period of time, so the worst case truth-telling margin that must be enforced by the RM is $\delta = \Delta - C/N/2 = 0.5\%$.

The belief of one client regarding the value of the reference report changes by at least $\bar{\gamma} = \underline{\gamma} = 20\%$ in the direction of the actual observation. The probability that the reference report contains 1 for Q_1 is: $Pr_1[1|1] = p_1 + (1-p_1)\bar{\gamma}$ if the client also received a response, or $Pr_1[1|0] = p_1 - (1-p_1)\bar{\gamma}$ if the client did not receive a response. Similar equations can be written for the probabilities $Pr_2[1|1]$ and $Pr_2[1|0]$ defining the beliefs regarding the value of Q_2 in the reference report. From (1), (2) and (3), Figure 2 presents the linear optimization problem that defines the minimum payments guaranteeing the truth-telling equilibrium. When $p_1 = p_2 = 90\%$, we obtain the payments: $\tau_1(1) = 0.064$, $\tau_1(0) = 0.680$, $\tau_2(1) = 0.025$, $\tau_2(0) = 0.225$, and an expected cost of 0.081. These rather high payments can be further decreased by an order of magnitude using a filtering mechanism: (i.e., the RM probabilistically selects the reports that will contribute towards estimating delivered quality). Similar to the payment mechanism, the filtering mechanism compares a report r with a set of peer reports, and specifies the probability of discarding r. In [9], we show

[7]The two criteria must be used together, since otherwise a service can achieve almost perfect availability by always returning the same information.

$$
\begin{aligned}
min \quad & E[Cost] = p_1^2\tau_1(1) + (1-p_1)^2\tau_1(0) + \\
& \quad p_2^2\tau_2(1) + (1-p_2)^2\tau_2(0); \\
s.t. \quad & Pr_1[1|1]\tau_1(1) > Pr_1[0|1]\tau_1(0) + \delta; \\
& Pr_1[0|0]\tau_1(0) > Pr_1[1|0]\tau_1(1) + \delta; \\
& Pr_2[1|1]\tau_2(1) > Pr_2[0|1]\tau_2(0) + \delta; \\
& Pr_2[0|0]\tau_2(0) > Pr_2[1|0]\tau_2(1) + \delta; \\
& Pr_1[0|0]\tau_1(0) > Pr_1[1|0]\tau_1(1) + \\
& \qquad p_2\tau_2(1) + \delta; \\
& Pr_1[0|0]\tau_1(0) > Pr_1[1|0]\tau_1(1) + \\
& \qquad (1-p_2)\tau_2(0) + \delta; \\
& \tau_1(1), \tau_1(0), \tau_2(1), \tau_2(0) \geq 0
\end{aligned}
$$

Figure 2: Linear optimization problem defining the payment mechanism.

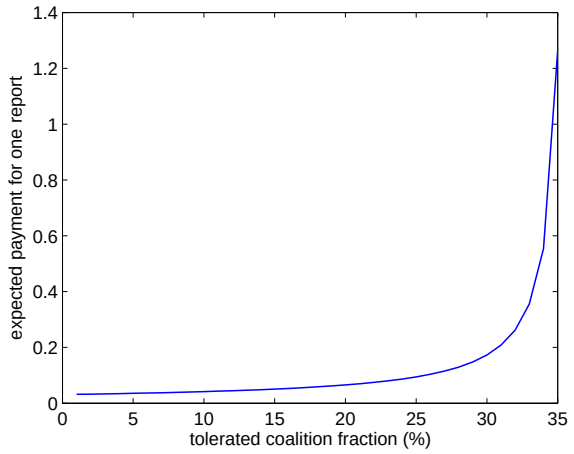

Figure 3: Expected cost of a payment mechanism that is robust against collusion.

that when designed together, the payment mechanism and the filtering mechanism enforce one another and consistently decrease the expected cost of the RM by a factor of 10. The combination of the two techniques brings down the expected payments to below 1% of the price of service, making them practical.

By modifying the optimization problem as suggested in Section 6, we obtain a payment mechanism that is also robust against coalitions that cover at most a fraction λ of the reports. The dependence of the expected cost on λ is plotted in Figure 3.

8. RELATED WORK

Our present work extends the line of research that argues for the use of reputation information in service markets.

RMs have emerged as efficient tools for service discovery and selection [25]. When electronic contracts cannot be enforced, users can protect themselves against cheating providers by looking at past behavior (i.e., the provider's reputation). Lie et al. [11] present a QoS-based selection model that takes into account the feedback from users as well as other business related criteria. The model is extensible and dynamic. In the same spirit, [10] proposes *verity*, a QoS measure that takes into account both reputation and the terms of the SLA. An interesting approach is proposed in [6]. The authors argue that the expectations of a client greatly influence the submitted feedback, and therefore both should be used when assessing the QoS of a provider. Both [14] and [1] propose concrete frameworks for service selection based on the reputation of the service provider. However, reputation-based selection gives only indirect incentives, as clients learn to avoid deceitful providers.

As opposed to the above solutions, we mainly use the feedback reported by the clients to substitute QoS monitoring. We believe that the information contained in the reports should be used directly and immediately to assess the *honesty* of the advertisement made by the provider. Moreover, this information should have direct repercussions on the gains of the provider through contractual penalties. In this way, providers get immediate incentives to exert effort.

The present paper extends our previous work [8] in several essential directions. First, we describe a detailed framework for reliable QoS monitoring based on client feedback and include the interaction protocols between the different actors in our environment. Second, we relax the assumptions behind our previous mechanism, and accommodate QoS monitoring along several dimensions. Third, we describe simpler payment systems that minimize the budget required by the RM and address the problem of collusion.

This paper also relates to the large body of research on monitoring and enforcing of electronic contracts ([29], [18], [13],[7],[3]).

Reliable information regarding the QoS of advertised services is essential for service selection and composition. In references [30, 31] the authors present AgFlow, a middleware for quality-driven service composition. In AgFlow, the QoS of web services is evaluated using an extensible multidimensional QoS model, and the selection of individual services aims at optimizing the QoS of the composite service.

Reference [27] introduces QoS-based selection of semantic web services, i.e., web services that provide well-defined, computer-interpretable semantics [16]. In reference [27] the authors describe a QoS model using the Web Service Modeling Ontology [28].

9. CONCLUSION

A service market based on SLAs between service providers and clients can only function well if advertised SLAs are credible. However, service providers may deviate from their advertised QoS in order to reduce the costs of service provisioning. Hence, QoS monitoring is essential, but neither service-side nor client-side monitoring can be trusted.

In this paper we presented a novel approach to achieve objective QoS monitoring by aggregating quality ratings from clients within a RM, which provides incentives for the clients to report honestly. The RM pays clients for submitting quality ratings, and the payments are designed such that lying generates expected losses that offset the potential benefits from misreporting.

10. ACKNOWLEDGEMENTS

We thank Romain Revol for his important contribution to the implementation of the QoS monitoring framework described in this paper.

11. REFERENCES

[1] B. Alunkal, I. Veljkovic, G. Laszewski, and K. Amin. Reputation-Based Grid Resource Selection. In *Proceedings of AGridM*, 2003.

[2] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement), Version 2005/09, http://www.ggf.org.

[3] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-Time Monitoring of Instances and Classes of Web-Service Compositions. In *Proceedings of ICWS 2006*, 2006.

[4] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the Uncertainty in Artificial Intelligence Conference (UAI)*, 2002.

[5] A. Dan, D. Davis, R. Kearney, A. Keller, R. P. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef. Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, 43(1):136–158, 2004.

[6] V. Deora, J. Shao, W. Gray, and J. Fiddian. A Quality of Service Management Framework Based on User Expectations. In *Proceedings of ICSOC*, 2003.

[7] Y.-J. Hu. Trusted Agent-Mediated E-Commerce Transaction Services via Digital Certificate Management. *Electronic Commerce Research*, 3, 2003.

[8] R. Jurca and B. Faltings. Reputation-based Service Level Agreements for Web Services. In *Service Oriented Computing (ICSOC - 2005)*, volume 3826 of *LNCS*, pages 396 – 409. 2005.

[9] R. Jurca and B. Faltings. Minimum payments that reward honest reputation feedback. In *Proceedings of the ACM Conference on Electronic Commerce*, Ann Arbor, Michigan, USA, June 11-15 2006.

[10] S. Kalepu, S. Krishnaswamy, and S. Loke. Verity; A QoS Metric for Selecting Web Services and Providers. In *Proceedings of WISEW*, 2003.

[11] Y. Liu, A. Ngu, and L. Yeng. QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of WWW*, 2004.

[12] H. Ludwig, A. Dan, and R. Kearney. Cremona: An architecture and library for creation and monitoring of WS-Agreements. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 65–74, New York, NY, USA, 2004. ACM Press.

[13] K. Mahbub and G. Spanoudakis. A framework for requirements monitoring of service based systems. In *Proceedings of ICSOC*, 2004.

[14] E. M. Maximilien and M. P. Singh. Toward Autonomic Web Services Trust and Selection. In *Proceedings of ICSOC*, 2004.

[15] M. Maximilien and M. Singh. A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[16] S. A. McIlraith and D. L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.

[17] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting Informative Feedback: The Peer-Prediction Method. *Management Science*, 51:1359 –1373, 2005.

[18] Z. Milosevic and G. Dromey. On expressing and monitoring behaviour in contracts. In *Proceedings of EDOC*, Lausanne, Switzerland, 2002.

[19] OWL-S. DAML Services, http://www.daml.org/services/owl-s/.

[20] I. Papaioannou, D. Tsesmetzis, and M. Roussaki, I. abd Anagnostou. A QoS Ontology Language for Web-Services. In *Proceedings of the International Conference on Advanced Information Networking and Applications (AINA 2006)*, 2006.

[21] M. P. Papazoglou and D. Georgakopoulos. Introduction: Service-oriented computing. *Communications of the ACM*, 46(10):24–28, Oct. 2003.

[22] D. Prelec. A bayesian truth serum for subjective data. *Science*, 306(5695):462–466, 2004.

[23] S. Ran. A Model for Web Service Discovery with QoS. *ACM SIGecom Exchanges*, 4(1):1–10, 2003.

[24] A. Sahai, V. Machiraju, M. Sayal, A. P. A. van Moorsel, and F. Casati. Automated SLA monitoring for web services. In *DSOM*, volume 2506 of *Lecture Notes in Computer Science*, pages 28–41. Springer, 2002.

[25] M. P. Singh and M. N. Huhns. *Service-Oriented Computing*. Wiley, 2005.

[26] W3C. Web services description language (WSDL) version 1.2, http://www.w3.org/TR/wsdl12.

[27] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A QoS-aware selection model for semantic web services. In *4th International Conference on Service Oriented Computing (ICSOC 2006)*, Chicago, USA, Dec. 2006.

[28] WSMO. Web Service Modeling Ontology, http://www.wsmo.org/.

[29] L. Xu and M. A. Jeusfeld. Pro-active Monitoring of Electronic Contracts. *Lecture Notes in Computer Science*, 2681:584–600, 2003.

[30] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *WWW*, pages 411–421, 2003.

[31] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Trans. Software Eng.*, 30(5):311–327, 2004.