

The Monitorability of Service-Level Agreements for Application-Service Provision

James Skene
Department of Computer Science
University College London
Malet Place
London, UK, WC1E 6BT
j.skene@cs.ucl.ac.uk

Jason Crampton
Information Security Group
Royal Holloway, University of London
Egham
Surrey, UK, TW20 0EX
Jason.Crampton@rhul.ac.uk

Allan Skene
Nottingham Clinical Research Limited
Nottingham Science and Technology Park
Nottingham, UK, NG7 2RH
a.skene@ncrl.co.uk

Wolfgang Emmerich
Department of Computer Science
University College London
Malet Place
London, UK, WC1E 6BT
w.emmerich@cs.ucl.ac.uk

ABSTRACT

Service-Level Agreements (SLAs) mitigate the risks of a service-provision scenario by associating financial penalties with aberrant service behaviour. SLAs are useless if their provisions can be unilaterally ignored by a party without incurring any liability. To avoid this, it is necessary to ensure that each party's conformance to its obligations can be monitored by the other parties. We introduce a technique for analysing systems of SLAs to determine the degree of monitorability possible. We apply this technique to identify the most monitorable system of SLAs governing timeliness for a three-role Application-Service Provision (ASP) scenario. The system contains SLAs that are at best mutually monitorable, implying the requirement for reconciliation of monitoring data between the parties, and hence the need to constrain the parties to report honestly while accommodating unavoidable measurement error. We describe the design of a fair constraint on the precision and accuracy of reported measurements, and its approximate monitorability using a statistical hypothesis test.

1. INTRODUCTION

In a commercial electronic service provision scenario, the client of the service assumes certain financial risks associated with the quality of the service. Before undertaking to receive the service, the client, possibly with the assistance of the provider, will have made certain assumptions about the future performance of the service, in particular for how long the service will continue to be available, and to what extent

the service will behave as advertised by the service provider. These assumptions have financial implications. The client will need to invest in integrating the service, on the assumption that the service is provided for long enough for them to recoup their investment, or otherwise justify the expense. The service as advertised will have some value to the client, so a sustained poor level of quality will imply a cost to the client.

Service-Level Agreements (SLAs), when part of service-provision contracts, are a mechanism for controlling these risks. By associating financial penalties with poor service performance, or early termination of the service, the client may have an improved chance of receiving compensation in either event.

This view of SLAs as a means by which to establish the liability of a party to pay a penalty to another party raises a number of requirements for the way in which SLAs should be written. Parties will generally not wish to pay penalties under any circumstances, and so are likely to contest the provisions of an SLA. If the original intent of the parties with respect to service provision is not properly captured by the SLA, or if a contest can successfully override the original intent, then the SLA has failed, because it did not mitigate the actual risk identified when it was written. We refer to the capacity of an SLA to withstand such contests as the *protectability* of the SLA.

A number of requirements for SLAs may be derived from the requirement that they be protectable, including that they be precise and understandable, an issue that we addressed in [10] by proposing that SLA languages be defined using an object-oriented meta-modelling formalism, which is both precise and familiar to technical users.

In this paper we consider the requirement that SLAs be *monitorable*. Monitorability implies that parties can oversee the behaviour of the service relevant to the SLA, or have it overseen on their behalf by parties that they trust. Without this ability it will be impossible for a party to assert that the SLA has been violated, and hence its provisions may be ignored by the service provider.

Monitorability for an SLA may be achieved by two means.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

First, the SLA may only place conditions on events that are intrinsically observable by the parties. Alternatively, any event may be constrained, provided that a technical solution can be devised to render it monitorable in a trustworthy manner. Such solutions possibly include the use of monitoring software executing on trusted computing platforms or otherwise tamper-proof hardware.

In this paper we restrict ourselves to the former approach, and offer an approach for the analysis of systems of SLAs to determine the monitorability of their constraints. Monitorability of an SLA is classified as unmonitorable, monitorable by one party, mutually monitorable by both parties, or arbitratable, when a third party trusted by both parties to the SLA can observe all relevant events, and hence arbitrate any dispute. Systems of SLAs have a similar categorisation, based on the monitorability of the least monitorable member SLA from the perspective of each party in the system.

We apply this analysis method to the electronic-service scenario shown in Figure 1. In this common scenario, supported by numerous middleware technologies, including RPC, J2EE, DCOM, .NET and web-services, the client submits a request to the service provider. The request is conveyed across a network operated by an Internet Service Provider (ISP). On arrival, the service performs some processing, which may involve interaction with the real world (for example, causing a parcel to be dispatched in the mail), and possibly manipulation of some information stored on the client’s behalf. Finally, a response is produced and sent to the client over the network. The scenario includes three roles, that of the client, the ISP and the service provider. A common constraint that the client will wish to associate with a penalty is the timeliness of their receipt of the response, relative to their dispatch of the request. We demonstrate that for all combinations of SLAs capable of insuring this constraint, only one system is monitorable. We show that this level of monitorability is also possible for SLAs involving networks administered by multiple ISPs, although we do not prove that no higher level is possible.

We next discuss the effect of this level of monitorability in a SLA to the requirements for handling measurement error and uncertainty in an SLA. In the case of SLAs that are monitorable, but not arbitratable, (the best case in our electronic-service scenario) the parties will need to negotiate the basis for the calculation of violations. To protect the agreement, this negotiation must concern the actual behaviour of the system. The parties must therefore be constrained to present evidence of the behaviour of the system that has been measured to a pre-agreed degree of accuracy. The design of such a constraint is non-trivial, as it must be fair to both parties, and it is also fundamentally not monitorable, as the true value of any event cannot be known with certainty. Therefore the constraint must be designed so that it can be approximately monitored using statistical methods. We describe our solution for the design of such constraints, and the means by which they may be approximately monitored by comparing an un-trusted log of measurements with presumed error characteristics with a trusted log with known error characteristics.

The theory presented in this paper has been applied in the design of the SLAng language for electronic service SLAs, which we briefly describe, and compare to previous work in this area.

The remainder of this paper is structured as follows: In

Section 2 we first motivate the concept of monitorability further. We then describe a general model for systems of SLAs and their classification. Throughout the paper we use the example of application-service provision. We describe an instantiation of our model for this scenario. We then describe the use of a depth-first search algorithm to perform analysis on models of SLAs and the results of this analysis for the ASP scenario. We also provide a short generalisation of these results to scenarios with multiple ISPs. In Section 3 we describe the design and approximate monitorability of a constraint on reporting accuracy appropriate for use in SLAs that are mutually monitorable but no better. In Section 4 we discuss related work, and in Section 5 we summarise.

2. MONITORABILITY

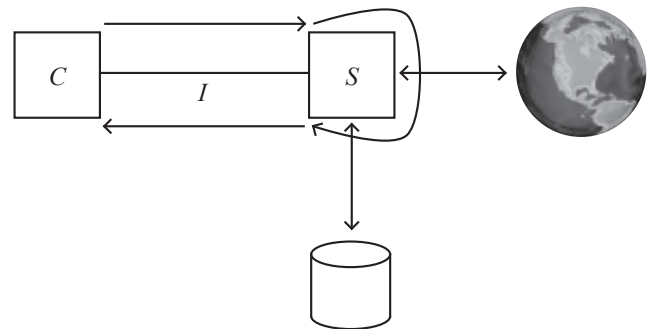


Figure 1: A three-party electronic service scenario

In this paper we will use the scenario described in the introduction and depicted in Figure 1 as both a running example and as the subject of our analysis.

From the client’s point of view a number of things could go wrong in this scenario. Let us consider the case that having submitted a request, no response is received by the client within some reasonable interval of time. The client complains to the service provider that a timely response was not received. The provider claims that no request was received, produces a log of requests as evidence supporting this claim, and directs the client to complain to the ISP who was responsible for conveying the request to the service. The ISP insists that the request reached the service provider and produces a log supporting this claim. Who can the client trust? Both the ISP and electronic-service provider have delivered easily fabricated evidence concerning an event, the delivery of the request at the service-provider’s interface, that the client was incapable of independently monitoring.

Let us assume that for their own reasons, the client chooses to mistrust the service provider, and requests that they enter into a service-level agreement. In this the client seeks to mitigate their own costs when the service fails to perform as expected, by receiving a penalty from the provider, and to give the provider a disincentive to poor performance. The client perceives that the problem with the service is a lack of availability due to an erratic maintenance regime on the part of the provider. The provider duly commits to provide 95% availability over the lifetime of the contract of which the SLA forms a part.

Over the period of the contract, the client uses the service frequently and frequently responses are not generated

following requests. At the termination of the agreement the client seeks compensation from the provider, who refuses to pay. The provider argues that although the service was unavailable when requests were made for which responses were not received, at all other times the service was available. Accumulating the microscopic intervals during which the (numerous) failed requests were being delivered and the service was admittedly unavailable still does not amount to 5% of the lifetime of the contract, and hence the provider need not pay. In this second example, the client has entered into an agreement which is, possibly implicitly, defined in terms of events that the client cannot directly observe, namely the passing of the service from an available to unavailable state, and vice versa. Since the client cannot observe these events, it must take the word of the service provider with respect to the availability of the service, or else pursue compensation with very little support from the original SLA.

These examples highlight two aspects of monitorability as an important requirement for SLAs. In both cases, the client became concerned with an event that they fundamentally could not observe: in the first example, the delivery of the request to the service; in the second, the transition of the service between availability states. In both cases, the motivation for this was because another event that they could observe, the delivery of the response to the client, failed to occur when expected. Had the client complained about this latter event, they would have had a stronger argument, because no party could convince them of a falsity concerning the event in question, but to whom should they complain? These examples highlight the importance of making SLAs with respect to monitorable events where possible.

An additional level of complexity is introduced by the potential unwillingness of parties to offer certain SLAs. For example, it would be unsafe for the service provider to offer guarantees on the performance of the service at the client's interface with the network. Although events there would be monitorable by the client, they would not be monitorable by the service provider, so the client could potentially exploit the provider. Moreover, the observed performance at that interface would also depend on the performance of the network, which is not under the control of the service provider. The electronic-service provider is likely to be unwilling to undertake to pay penalties if the performance of the network is inadequate.

Considerations of this kind for a particular scenario beg the question: is any system of SLAs possible to guarantee a particular requirement in which all SLAs are safe and monitorable? In the next sections we develop an abstract mathematical model of such scenarios and describe how it may be refined and permuted to answer this question, and provide other insights, for a particular scenario. We continue the running example of application service provision and provide an analysis of the monitorability of systems of SLAs in that scenario.

2.1 Modelling systems of SLAs

We assume that two or more parties participate in some form of interaction comprising a sequence of actions each performed by one of the participants. Examples include electronic service provision, resource provisioning in virtual organisations, etc. Particular parties may have certain expectations about the execution of particular actions. For example, a party may specify a requirement that these ac-

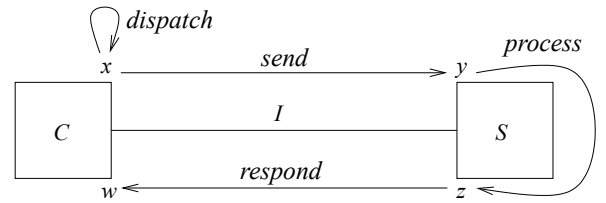


Figure 2: An interaction model for application service provision showing actions and their associated events

tions are executed within a certain amount of time, as in the case of service provisioning. Another participant may agree to pay penalties to this first party if these requirements are not met.

DEFINITION 1. *An interaction between two or more entities belonging to a set of participants P is modelled as a sequence of actions, A . Each action a is associated with an actor $\alpha(a) \in P$, the party that may perform the action.*

Figure 2 depicts the interaction model for the electronic service scenario. Interactions with the real world and any database have been elided. Formally, the model is expressed as follows:

In the three party scenario: C is the client, I the ISP and S the service provider. $P = \{C, I, S\}$

The client occasionally takes action to invoke the service by dispatching requests into the network. The ISP conveys requests through the network, eventually delivering them to the server (operated by the service provider). Having received a request, the server performs some service and injects the result back into the network. The ISP is then responsible for the delivery of the result to the client. The actions are $A = \{\text{dispatch}, \text{send}, \text{process}, \text{respond}\}$, and responsibility is allocated as follows: $\alpha(\text{dispatch}) = C$, $\alpha(\text{send}) = I$, $\alpha(\text{process}) = S$, and $\alpha(\text{respond}) = I$.

DEFINITION 2. *An action a may give rise to any number of events $\epsilon(a) \in E$, where for all pairs of actions a and a' , $\epsilon(a) \cap \epsilon(a') = \emptyset$.*

The events modelled for a given scenario depend on the requirements of interest to the parties. We will now restrict our analysis of application service provision to systems of SLAs governing the timeliness of service provision. We therefore define events corresponding to the completion of each of the events in the scenario $E = \{x, y, z, w\}$ in relation to our actions such that $\epsilon(\text{dispatch}) = \{x\}$, $\epsilon(\text{send}) = \{y\}$, $\epsilon(\text{process}) = \{z\}$, and $\epsilon(\text{respond}) = \{w\}$. These events are chosen because service provision should be deemed to begin when a request has been fully submitted, and end when a response has been fully received. It is easy to see that this simple model meets the requirement that no event is the result of more than one action.

Our monitorability analysis will be supported by the notion that particular events are only visible to a subset of the parties in the interaction:

DEFINITION 3. *Each event $e \in E$ has a set of respondents $\rho(e) \subseteq P$, the parties that may observe the event. It is assumed that for all $e \in \epsilon(a)$, $\alpha(a) \in \rho(e)$.*

The events in our model occur at network interfaces, so we define the respondents for each event as follows: $\rho(x) = \rho(w) = \{C, I\}$, and $\rho(y) = \rho(z) = \{I, S\}$. Clearly, each event is visible to the actor responsible for that action that causes it.

We now require a means by which to model requirements over events:

DEFINITION 4. *Events may have attributes. A set of observations, O , may be defined over these attributes, with each proposition $o \in O$ pertaining to a subset of events $\pi(o) \subseteq E$. These observations can be considered to be logical predicates concerning the values of attributes of observed events.*

Latency constraints in SLAs will potentially place restrictions over the timing of the events x, y, z and w . We may now state that occurrence time is an attribute of the events, and define observations that capture the requirements of the parties and the constraints that we may wish to include in SLAs.

Specifically, we wish to express the client's requirement that w occurs within some interval following x . We denote the observation that this occurs in abstract as $w - x$, where the specific size of the interval is not a concern. This constraint can potentially be achieved by constraining the relative times of events occurring between w and x , the intuition being that the overall time taken to complete a request is acceptable if the times taken to complete each part of the request are also acceptable. The total set of observations with which we will be concerned is hence $O = \{y - x, z - x, w - x, z - y, w - y, w - z\}$.

The mapping π from observations to the events over the attributes of which the observations are defined is obvious in this case from the naming of the observations, e.g. $\pi(y - x) = \{y, x\}$.

Clearly, if observations are predicates over a scenario, then the truth values of observations are potentially related. In the ASP case, we can state that $w - x$ will always hold if $w - z, z - y$ and $y - x$ hold, amongst other relationships.

DEFINITION 5. *Observations are related by an entailment relation, $\models_{\subseteq} 2^O \times O$, where $S \models o$ is interpreted as stating that o is always true when all observations in S are true.*

To more conveniently describe relationships between observations, a minimal dependency mapping D may be defined based on the entailment relationship. We say $S \subset O$ entails observation o if $S \models o$ and we say S is minimal if for all $S' \subset S, S' \not\models o$. Hence $D(o) = \{S \subseteq O - \{o\} : S \models o, \text{ and } S \text{ is minimal}\}$.

In our scenario, the dependency mapping may be enumerated as follows:

$$\begin{aligned} D(w - x) &= \{\{w - z, z - y, y - x\}, \\ &\quad \{z - x, w - z\}, \\ &\quad \{w - y, y - x\}\} \\ D(w - y) &= \{\{w - z, z - y\}\} \\ D(z - x) &= \{\{z - y, y - x\}\} \\ D(w - z) &= \emptyset \\ D(z - y) &= \emptyset \\ D(y - x) &= \emptyset \end{aligned}$$

DEFINITION 6. *A party $c \in P$ may impose certain requirements on the execution of a sequence of actions. These requirements are modelled by associating the party with the observation that they wish to hold in a pair $(c, o) \in R$.*

The client's latency requirement is the fundamental requirement addressed in this analysis. Hence $R = \{(C, w - x)\}$.

DEFINITION 7. *A party p can provide an SLA to insure any requirement. The SLA states that the party with the requirement will receive compensation from p if the requirement is not met. SLAs are modelled as a pair $(p, (c, o)) \in S$ where S is the set of SLAs in a particular scenario and (c, o) represents a requirement.*

For example, I could offer C an SLA matching C 's requirement: $(I, (C, w - z))$. Any combination of parties with guarantees is possible, so for example, C might also offer S an SLA of the form $(C, (S, w - y))$.

Having now established definitions for SLAs and the observability of events in our model, we may begin to define the levels of monitorability possible in a given model.

Although parties may not be able to observe an event themselves, they may have the event reported to them by a party that they trust. However, we suggest the conservative restriction that a party should not be trusted to report on an event if they have a financial interest in that event. We state that a party has a financial interest in an event if they provide or receive an SLA that insures an observation to which the event is pertinent. The interest arises from the desire of the client to receive penalty payments, and the desire of the provider to avoid paying them.

DEFINITION 8. *We say a party p may reliably report on event e if there is no SLA $(p, (c, o)) \in S$ or SLA $(c, (p, o)) \in S$ for any other party c such that $e \in \pi(o)$. The set of parties in a given scenario that may reliably report on an event e is denoted $\tau(e) \subseteq P$.*

Provided that a party may reliably report on an event, another party may choose to trust them:

DEFINITION 9. *A party q may choose to trust a party p to report on an event e , if p can reliably report on e . For each party p and each event e , we define $\tau(p, e) \subseteq \tau(e)$ to be the set of participants that p trusts to report on the event e .*

DEFINITION 10. *We say a party p may monitor an event e if $p \in \rho(e)$ or there exists a party $q \in \tau(p, e)$ who can monitor the event independently of p and who p trusts to monitor that event. We recursively define a generic mapping $\mu_M(e) = \{p \mid p \in M \wedge (p \in \rho(e) \vee \exists q.(q \in \tau(p, e) \wedge q \in \mu_{M - \{p\}}(e)))\}$ of all parties in a subset $M \subseteq P$ that can monitor an event e . We may therefore define $\mu(e) = \mu_P(e)$ as the set of all parties that can monitor an event e .*

Note that the set of parties who may be trusted depends on the set of SLAs issued. Therefore, in our scenario, C can only choose to trust I or S to report on events y and z (which C cannot monitor directly) if I or S do not offer or receive any SLAs related to these events.

DEFINITION 11. *Given an SLA $(p, (c, o))$, a party q can monitor the SLA if it can monitor all events $e \in \pi(o)$.*

For example, to monitor an SLA related to the observation $y-x$ a party must be able to monitor both y and x , in order to determine the arrival and departure times of a request. If S issues $s_1 = (S, (C, z-x))$ to C and I also offers $s_2 = (I, (C, w-z))$ to C , and no other SLAs are made, then s_1 will possibly be monitorable to S , because S can directly observe z and might choose to trust I to report on x , because I offers no SLAs pertinent to x . C on the other hand cannot monitor s_1 because it cannot trust either S or I to report on z . Similarly s_2 is directly monitorable by I but cannot be monitored by C for the same reason as s_1 .

Monitorability of an SLA is particularly desirable for a party that is the client or the provider of the SLA, as that party can know what penalties should be paid. In a fair scenario it would be desirable for both parties to be able to monitor the SLA, since then neither party could cheat the other without the other party being aware of it.

DEFINITION 12. We say an SLA, $(p, (c, o))$, is mutually monitorable if p and c can monitor the SLA.

Supposing that S offered C , $s = (S, (C, z-y))$ and I and C offered nothing. s would be mutually monitorable if C trusted I to report on z and y .

Ideally an SLA would be monitorable by a third party, trusted by both client and provider to report honestly. Since the third party was trusted, it could be relied upon to arbitrate disputes between the client and provider.

DEFINITION 13. If there exists a third party t such that for all $e \in \pi(o)$, $t \in \tau(c, e)$ and $t \in \tau(p, e)$ then we say the SLA is arbitratable by t , since both parties trust t to monitor the SLA.

Supposing that S offered C , $s = (S, (C, y-x))$ and I and C offered nothing. s would be arbitratable by I providing that I was trusted by both C and S .

In the preceding example, and others above, there seems to be something intuitively wrong with the SLAs offered, in that parties attempt to insure the behaviour of actions that they do not themselves perform. To avoid this we introduce a notion of guarantees, and subsequently characterise safe SLAs as being those that rely either on guarantees of which the provider is capable, or subordinate guarantees acquired from elsewhere.

DEFINITION 14. A party $g \in P$ may, by their actions, be able to guarantee that an observation holds. Guarantees are modelled like requirements as a pair $(g, o) \in G$. The guarantor must be able to monitor all events pertinent to the observation. The guarantor must also perform actions that cause a subset of the events pertinent to the observation, i.e. there must exist $e \in \pi(o)$ and $a \in A$ such that $g = \alpha(a) \wedge e \in \epsilon(a)$.

Recalling that observations are predicates over the attributes of events, it is clear that to guarantee that an observation holds a guarantor must meet the two conditions defined in the observation. By causing some pertinent events, the guarantor can vary the values of the attributes of these events, thereby causing the observation to hold. However, to determine how this should be done, the party must also be able to determine the values of the attributes of the other pertinent events. In general, a party's capacity to guarantee observations may therefore depend on what events are

monitorable to that party, and hence on the SLAs that are offered in a scenario.

In our example, the ISP and the service provider can guarantee several observations regardless of what SLAs are offered. The ISP can control the time taken to deliver the request to the server, once it has been received by the network. Note that the ISP performs the action that causes y and can always monitor x directly. The service provider controls the time taken to perform processing once the server has received the response. The ISP again controls the time taken to deliver the response once it has been received by the network. The following guarantees are therefore included in the model: $G = \{(I, y-x), (S, z-y), (I, w-z)\}$.

DEFINITION 15. An SLA $(p, (c, o))$ is safe to issue if p can guarantee o , i.e. if $(p, o) \in G$, or p can obtain an SLA $(q, (p, o))$ from a second party q , or if these conditions can be satisfied for all observations in any set of observations upon which o depends. I.e. for all $o' \in S$ where $S \in D(o)$, p can either guarantee o' or obtain an SLA for o' from a second party.

If an SLA is safe to issue, the provider p may be liable to pay penalties when requirements are not met due to the actions of other parties, but will also receive penalties, which, appropriately negotiated, will obviate their risk. In the case of an SLA that is unsafe to issue, the provider may have to pay penalties due to the actions of other parties without receiving compensation themselves.

Having defined and motivated a set of characteristics for individual SLAs, we may also describe systems of SLAs as follows:

DEFINITION 16. A system of SLAs may be characterised from the point of view of a party contained in it as satisfactory if all requirements of the party are met. A system is satisfactory overall if it is satisfactory for all parties that it contains.

DEFINITION 17. A system of SLAs is safe from the point of view of a party if all SLAs that the party issues are safe to issue. A system of SLAs is safe overall if it is safe for all parties.

DEFINITION 18. A system of SLAs is monitorable from the point of view of a party if all SLAs issued or received by the party are monitorable by the party. A system of SLAs is monitorable overall if it is monitorable by all parties.

DEFINITION 19. A system of SLAs is arbitratable if all the SLAs it contains are arbitratable.

Finally it may be desirable in an analysis to rule out the following types of system of SLAs:

DEFINITION 20. A system of SLAs S may also be characterised as redundant if there exists $S' \subset S$ and S' is both satisfactory and safe.

DEFINITION 21. A system of SLAs may be characterised as reciprocal if it contains two SLAs $s_1 = (p, (c, o))$ and $s_2 = (c, (p, o))$. In other words, two parties exchange SLAs with respect to an observation.

2.2 Monitorability analysis

A particular system of SLAs may be characterised as described in the previous section, depending on the degree of satisfaction, safety and monitorability afforded to its parties, and the possible redundancy or reciprocity of its SLAs.

Searches for systems of SLAs with particular requirements are possible by keeping most of the model constant, then varying the set of SLAs used. For example, we might ask for a given scenario: what sets of SLAs are safe, satisfactory and monitorable?

To identify sets of SLAs possessing a specific set of the characteristics defined in the previous section, we could in principle generate all combinations of SLAs, classify each, and then accept or reject systems of SLAs according to their classification.

The maximum number of possible SLAs in a given scenario is $|O| \times |P| \times (|P| - 1)$.

The number of combinations of SLAs is therefore $2^{|O| \times |P| \times (|P| - 1)}$. This is potentially a very large number, suggesting that a more intelligent strategy for identifying useful combinations of SLAs is needed.

Depth-first search is an appropriate technique for finding sets of SLAs. We propose the following algorithm for generating and testing sets of SLAs, presented in pseudo-code:

```

procedure DEPTH_FIRST()
begin
    return DEPTH_FIRST({}, {})
end

procedure DEPTH_FIRST(tentative, tried)
begin
    result := {}
    next := FILTER(tentative,
        GENERATE(tentative, tried))
    for each n in next
        result := result union
            DEPTH_FIRST({ n } union tentative, tried)
        tried := tried union { n }
    if ACCEPT(tentative) then
        result := result union { tentative }
    tried = tried minus next
    return result
end

procedure GENERATE(tentative, tried)
begin
    result := {}
    for each c in P
        for each p in P
            for each o in O
                if not c = p then
                    s := (c, (p, o))
                    if not s in tentative and
                        not s in tried then
                        result := result union { s }
end

procedure ACCEPT(tentative)
begin
    return true
end

```

```

procedure FILTER(tentative, next)
begin
    return next
end

```

The algorithm as presented will generate all possible combinations of SLAs. Note that the algorithm maintains a tentative set of SLAs that may be added to the result if they pass a test defined by `ACCEPT`. However, `GENERATE` first attempts to generate candidate SLAs to add to this set, these are filtered by `FILTER`, and then `DEPTH_FIRST` is recursively called to investigate each resulting tentative set. A set of SLAs that have already been tried is also maintained to avoid repeatedly trying to add the same SLAs in a different order.

Potential efficiency benefits of the approach rely on re-defining the heuristic operation `FILTER` to focus the search of the algorithm. `ACCEPT` may also be redefined to narrow the selection criteria for sets of SLAs, for example to accept only sets of SLAs with a minimum level of monitorability.

A possible rewrite for `FILTER` uses the tentative set to identify requirements that are not yet satisfied if the SLA is to be safe and satisfactory, and eliminates or defers the consideration of SLAs that do not have the potential to contribute to the requirements. The dependency relationship can guide this. `FILTER` may also remove all SLAs from the extension set if `ACCEPT` will reject the tentative set and all supersets, as will be the case if `ACCEPT` rejects redundant or reciprocal sets.

Note that in its current form the algorithm is no more efficient than any other method for generating and testing all possible combinations of SLAs. In the worst case for a particular scenario and set of criteria, all systems of SLAs will meet the criteria. In general therefore no algorithm can have complexity less than $\mathcal{O}(2^{|O| \times |P|^2})$.

It would be desirable in the future to prove formally that this algorithm always terminates and generates all possible combinations of SLAs for a given scenario. It would also be desirable to be able to prove, for a particular re-implementation of `ACCEPT` and `FILTER`, that the algorithm finds all sets of SLA with characteristics acceptable to the criteria defined in `ACCEPT`. That it will only find such SLAs is self-evident, as `ACCEPT` must return true before a tentative set can be added to the result.

2.3 SLAs for the ASP scenario

In our example scenario $2^{3 \times 2 \times 6} = 2^{36} \sim 6.9 \times 10^{10}$ distinct sets of SLAs are possible.

We are interested in the sets with the following properties: safety, satisfaction, non-redundancy, non-reciprocity, sets in which the client issues no SLAs (non-client), monitorability and arbitratability. We will also be interested in sets with combinations of these properties.

Some of these sets can be discovered with the depth-first search algorithm described in the previous sections. Others do not permit sufficient narrowing of the search space to render this approach feasible, but the number of these sets can be determined analytically.

In this paper we present analytical solutions for the total number of satisfactory sets, the total number of non-reciprocal sets, and the total number of non-client sets.

The number of satisfactory sets can be determined by considering the SLAs that must be offered to the client to satisfy

Sat	Safe	Non-red	Non-rec	Non-client	Mon	Arb	Sets considered	Solutions
-	-	-	-	-	-	-	-	$\sim 6.9 \times 10^{10}$
✓	-	-	-	-	-	-	-	$\sim 6.6 \times 10^{10}$
-	-	-	✓	-	-	-	-	$\sim 3.9 \times 10^8$
-	-	-	-	✓	-	-	-	$\sim 1.7 \times 10^7$
✓	✓	✓	-	-	-	-	16001	281
✓	✓	✓	✓	-	-	-	7696	122
✓	✓	✓	✓	-	✓	-	7696	1
✓	✓	✓	✓	✓	-	-	3571	34
✓	✓	✓	✓	✓	✓	-	3571	1
✓	✓	✓	✓	✓	✓	✓	3571	0

Table 1: Results of monitorability analysis for ASP scenario, with performance of depth-first search algorithm

its requirement $w - x$. These must insure at minimum $w - x$ or any dependency set for $w - x$, which in this case includes all other observations. A total of 36 possible SLAs may be offered in this scenario. However, we are only interested in those that offer guarantees to the client, of which there are 12 (either I or S may offer an SLA for any observation to C). We can therefore determine the total number of satisfactory sets of SLAs by determining how many combinations of these 12 SLAs result in the satisfaction of the client’s requirement, then multiplying this number by the number of combinations of the SLAs with which we are not concerned.

There are $2^{12} = 4096$ combinations of SLAs that may be offered to the client. A truth-table inspection of these combinations reveals 3927 to be satisfactory. The total number of satisfactory SLAs is therefore:

$$2^{24} * 3927 \approx 6.6 \times 10^{10}$$

A Java implementation of the truth-table analysis for satisfaction is available online [1].

The number of non-reciprocal SLAs can be determined straightforwardly. Each possible SLA has a reciprocal SLA with which it should not appear, making 18 pairs, any one of which should not appear. There are therefore $2^{36} * (3/4)^{18} = 3^{18} \sim 3.9 \times 10^8$ non-reciprocal sets of SLAs.

We may wish to make the restriction that the client does not offer any SLA. This will reduce the space of the search to $2^{24} \sim 1.7 \times 10^7$

We employed a Java implementation of the depth first search algorithm to discover non-redundant sets of SLAs, and those with minimal levels of monitorability. Note that non-redundant sets of SLAs are by definition both safe and satisfactory.

The non-redundancy, non-reciprocity and non-client constraints all have the effect of considerably limiting the size of the search space when employing depth first search. This is because a redundant or reciprocal set, or a set containing a client SLA cannot be improved by the addition of SLAs.

A summary of our experiments and analytical results is shown in Table 2.3. The Java implementation is available under an open-source licence for inspection and modification [1].

The most significant result of this analysis is that in exactly one of these arrangements can all sets of SLAs be monitored by the parties to them. This is true whether or not we permit the client to offer SLAs.

In this scenario, for a system of SLAs to be safe and satis-

factory both S and I must issue SLAs supported by guarantees contributing to C ’s requirements. Hence all parties will be financially involved in every contractual situation, and no party can be trusted to report any events that occur remote from another party. Therefore monitoring is only possible directly, and hence only SLAs between adjacent parties can be mutually monitored, namely, contracts between C and I , and I and S . Only one scenario meets this requirement. It consists of the contracts $(I, (C, w - x))$ and $(S, (I, z - y))$. The ISP guarantees that the service will perform correctly across its interface with the client. It is capable of guaranteeing that the request reaches the server in a timely fashion, and that any response makes it back in time. To fully guarantee the round-trip time of the service the ISP must only obtain a guarantee from the service provider that the service will complete in good time.

That no arrangement can be arbitrated is obvious without applying the search algorithm. Because all parties in the scenario must be involved in contracts to satisfy C ’s requirement, no financially independent third party can be present to observe any interaction.

That the scenario is only monitorable in one system of SLAs is a highly significant result. This system of contracts requires the ISP to offer guarantees on the received quality of an electronic service at the interface to the client, effectively forcing the ISP to act as a re-seller of application services, a business model not adhered to in practice today. Service constraints will be required at both the interface to the client and the interface to the service. The guarantees required in both places will be of the same form, although the constraints at the service interface will need to be tighter to accommodate any delay in the network whilst guaranteeing requirements at the client interface. Therefore to achieve monitorable end-to-end QoS guarantees for ASP, only one type of SLA language need be used. There is no need for a separate language to describe network QoS, for example. However, ISPs will have to offer ASP SLAs.

2.4 Multiple ISPs

The results of the previous section may be generalised to scenarios including multiple ISPs and clients distributed in the network. Clearly a sequence of SLAs can be established between each client and the service, such that each SLA is made between two adjacent parties, one serving as the client and the other as the service provider. This situation is shown in Figure 3. In the figure the original service S_0 is embedded in a network I_1 . I_1 can hence provide the service

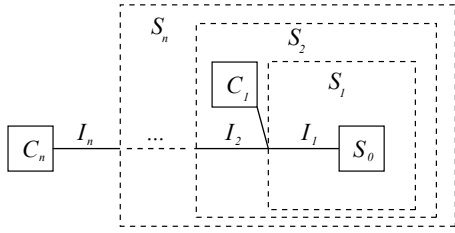


Figure 3: Monitorability is possible for ASP SLAs across chains of ISPs by regarding ISPs encapsulating the service as service providers, hence $I_i = S_i$ for $i > 0$. Clients may be embedded in any network

to C_1 . It can also exchange requests and responses with the linked network I_2 . Multiple networks are linked to provide a path to client C_n .

Clearly a system of SLAs that is monitorable can be provided for C_1 , as this is the same situation as analysed in the previous section. That a system can also be provided for clients embedded in any network I_i , where $i = 2 \dots n$ can be seen by considering the fact that in exchanging requests and responses with I_2 , I_1 is behaving exactly like a service embedded in network I_2 . This is indicated using the dashed box which re-identifies I_1 and S_0 as S_1 . Therefore provided I_1 and I_2 make an SLA insuring the timeliness of responses at their mutual interface, then it will also be possible for I_2 to offer such an SLA to any client embedded in their network. Clearly the SLA between I_1 and I_2 will be mutually monitorable, since it need only concern events occurring at the parties' mutual interface. This argument applies inductively for any number of network links, so it will be possible to provide a chain of monitorable SLAs to insure the timeliness requirements of client C_n . In general, clients may be embedded in any network provided a path to the original service exists, and SLAs are can be made at each network boundary.

This analysis establishes that when multiple ISPs are involved in delivering the service, then systems of SLAs exist that are at least monitorable. What has not yet been established is whether in this scenario arbitratability can be achieved for some or all parties.

3. RECONCILIATION AND ERROR

Fair administration of SLAs requires accurate data concerning the behaviour of services. In at least one important case this data must be provided by a party that is not inherently trustworthy, but whose reporting can be monitored. This occurs whenever an SLA is monitorable but not arbitratable. In this case, the client and provider of the service must agree a reconciled account of service performance on which to base the calculation of violations and hence penalties. This is because the parties have no third-party upon whom to rely, and neither party can fully establish the validity of their own account of the service performance by technical means alone.

In principle, the agreed account need bear no resemblance to the real behaviour of the service, providing the parties agree. However, in the event of a disagreement between the parties, honest parties who are concerned with the real behaviour of the system will wish to be able to argue that

their counterpart has neglected their responsibilities. It is therefore necessary to include in such SLAs an obligation that parties report the behaviour of the service accurately.

The inevitable presence of error and uncertainty in any measurement of a physical system raises two problems in the face of such a requirement. First, how can such an obligation be formulated to permit a tolerable degree of error in reporting, at the same time penalising higher levels of error, whilst respecting the right of the client to vary their utilisation of the service, and both parties to conceal details of their monitoring solutions. Second, how can conformance to such a constraint be checked, given that the true performance of the system cannot be determined with total certainty.

In the following sections we formulate such a constraint and explain how a statistical hypothesis test can be used to tell with some degree of confidence whether the constraint has been violated. Since complete confidence in this result is not possible, we introduce the term 'approximately monitorable' to refer to this type of constraint.

While describing the constraint we also give advice as to how parameter values may be chosen so that the parties may be confident that they can meet their obligations, assuming that they understand the error characteristics of their monitoring process.

The constraint chosen requires the parties to provide minimal information concerning the error characteristics, and by implication the implementation of their monitoring solutions. We also investigate the degree of fraud that is possible assuming that a party has perfect knowledge of their error characteristics, but need not reveal this information.

3.1 Accuracy constraint

The constraint that we have chosen is designed to restrict the probability that a reported measurement lies outside a specified interval of the true value. A log consists of a sequence of measurements X_i of event values μ_i where $i = 1, \dots, n$. Our constraint therefore guarantees of the reported values that:

$$pr(|X_i - \mu_i| > e) \leq 1 - c \quad (1)$$

Where e is the specified error margin, and c a specified confidence in the measurement. μ_i , the true value of the event is always unknown.

Since a party is not required to report any more accurately than is guaranteed by the SLA, we assume the probability p of a measurement in a given log being accurate is:

$$p = pr(|X_i - \mu_i| > e) = 1 - c \quad (2)$$

A measurement is erroneous if it falls outside of the error interval centered on the true value. We wish to limit the number of erroneous values in any given log. Although the true value μ_i of a measurement can never be known, it can and must be referred to in our accuracy constraint as the basis for defining accuracy.

Let d denote the number of erroneous measurements in a particular log. We wish to prohibit the reporting of logs containing improbably large numbers of erroneous values.

Assuming that a party reports honestly, using a monitoring system with the above error characteristics, the probability of a log of size n containing d erroneous measurements is given by the binomial distribution:

$$pr(d) = \binom{n}{d} p^d (1-p)^{n-d} \quad (3)$$

For a particular size of log, n , and choice of SLA parameters e and c , we therefore determine an upper bound on d , d_0 such that:

$$\sum_{d=d_0+1}^n \binom{n}{d} p^d (1-p)^{n-d} < \alpha \quad (4)$$

Where α is a parameter specified in the SLA. In other words, we wish to bound the likelihood that the log contains more than d_0 erroneous values.

The constraint therefore chosen is for a given log:

$$d \leq d_0 \quad (5)$$

The formulation of the constraint is analogous to a statistical hypothesis test, in which d is the test statistic, and the null hypothesis is that the log is honest. α is equivalent to the type I error rate for the test – the probability that an honest log is rejected as dishonest by the constraint.

3.2 Approximate monitorability

It is not possible to determine with certainty that a party has conformed to the constraint described in the previous section, because determining the number of erroneous values d requires the true values of the events μ_i to be known, and they cannot be known with certainty.

However, if a contract is monitorable then then all parties to the contract will be able to obtain trusted measurements of all events pertinent to the contract. Hence, for any of these parties, it will be possible to approximately monitor an untrusted party's conformance to the accuracy constraint by comparing the untrusted log that they produce, measurements X_i , with a trusted log, measurements Y_i , that they have obtained themselves (possibly with the help of trusted third parties).

The approximate monitoring of the accuracy constraint is achieved via a statistical hypothesis test. The null hypothesis is that the untrusted party has produced an honest log. The alternative hypothesis is that untrusted party is cheating, or otherwise failed to conform to the accuracy constraint.

H_0 : Contractor is honest

H_1 : Contractor is unable or unwilling to conform to the accuracy constraint.

We wish to detect erroneous values. It is not possible to compare X_i to μ_i , but we can compare it to Y_i . Y_i is trusted, and we assume that some characteristics of its error distribution are well understood. In particular we assume that its measurements are at least as accurate as required by the accuracy constraint. Therefore, if the absolute difference between the two logs is greater than $2e$ then the probability that X_i lies further than e from the true value μ_i is related to the confidence we have that Y_i lies within the error interval of the true value, which in the worst case will be given by c . The test statistic chosen for our hypothesis test is therefore d' , the number of cases where $|X_i - Y_i| > 2e$.

Large values of d' will favour H_1 . Similar to the formulation of the constraint, it is therefore a matter of comparing this statistic to a threshold value d'_0 such that:

$$pr(d' > d'_0 | H_0 \text{ true}) = \alpha \quad (6)$$

In order to determine d'_0 we need the sampling distribution for d' when H_0 is true. This is again given by the binomial distribution:

Let:

$$p' = pr(|X_i - Y_i| > 2e | H_0 \text{ true}) \quad (7)$$

The sampling distribution of d' is hence:

$$pr(d') = \binom{n}{d'} p'^{d'} (1-p')^{n-d'} \quad (8)$$

The problem, therefore, is to find an expression for p' in terms of e and c . This could be determined exactly if the error distribution for each party were known, but in practice we can only assume (under the null hypothesis) that the distributions of the parties conform to the parameters given in the accuracy constraints. The best that is therefore possible is an upper bound for p' .

The Chebychev inequality [14] states that given any random variable X where $E(X) = \mu$ and $Var(X) = \sigma^2$ then:

$$pr(|X - \mu| > k\sigma) \leq \frac{1}{k^2} \quad (9)$$

Therefore, under the null hypothesis:

$$pr(|X_i - \mu_i| > k\sigma) \leq \frac{1}{k^2} \quad (10)$$

The untrusted party has agreed that the error in their log will obey the rule

$$pr(|X_i - \mu_i| > e) \leq 1 - c \quad (11)$$

Because we can assume under the null hypothesis that the untrusted party is honest, but no better, to obtain our upper bound for p' , we must assume:

$$pr(|X_i - \mu_i| > e) = 1 - c \quad (12)$$

Equating the parameters of the constraint with those of the Chebychev inequality, we obtain:

$$k\sigma = e \quad (13)$$

And:

$$1 - c = \frac{1}{k^2} \quad (14)$$

Resulting in the following worst-case relationship between the standard deviation of the error distributions and the parameters of the constraint:

$$\sigma = e\sqrt{1-c} \quad (15)$$

Or:

$$e = \frac{\sigma}{\sqrt{1-c}} \quad (16)$$

Now consider $X_i - Y_i$. We make the following assumptions:

1. $E(X_i - \mu_i) = E(Y_i - \mu_i) = 0$
2. The variance of the trusted log, σ_Y^2 is known.

The first assumption, that the measurements are unbiased, is reasonable, because any systematic bias on the part of either party will easily be detected and rectified when the two logs are administered, and does not represent a clear intent of the party to deceive their peer. Occasionally both parties may suffer from similar biases, which may hence be overlooked. This will not be problematic from the point of view of obtaining an agreement between the parties, and may be rectified if selected later.

The second assumption reflects the fact that the trusted log has been obtained via a measurement process the error characteristics of which are known to some degree. We have already assumed that the measurement process gives results conforming to the accuracy constraint on the untrusted log. Therefore in the if nothing else is known we may assume $\sigma_Y^2 = \sigma$.

Since in the worst case according to the null hypothesis the variance of the untrusted log is σ then the variance of $X_i - Y_i$ will be:

$$\text{Var}(X_i - Y_i) = \sigma^2 + \sigma_Y^2 \quad (17)$$

By introducing a constant r such that:

$$\sigma^2 + \sigma_Y^2 = r\sigma^2 \quad (18)$$

We may state our assumptions in the form:

$$E(X_i - Y_i) = 0, \quad \text{Var}(X_i - Y_i) = r\sigma^2 \quad (19)$$

Therefore, by Chebychev again

$$\text{pr}(|X_i - Y_i| > k\sqrt{r}\sigma) = \frac{1}{k^2} \quad (20)$$

The original relationship still holds:

$$p' = \text{pr}(|X_i - Y_i| > 2e) = \text{pr}\left(|X_i - Y_i| > 2\left(\frac{\sigma}{\sqrt{1-c}}\right)\right) \quad (21)$$

Equating the parameters, and cancelling the σ terms:

$$k\sqrt{r} = \frac{2}{\sqrt{1-c}} \quad (22)$$

$$k = \frac{2}{\sqrt{r}\sqrt{1-c}} = \frac{2}{\sqrt{r-rc}} \quad (23)$$

And:

$$p' = \text{pr}(|X_i - Y_i| > 2e) = \frac{1}{k^2} = \frac{1}{\left(\frac{2}{\sqrt{r-rc}}\right)^2} = \frac{r-rc}{4} \quad (24)$$

d'_0 can hence be determined for given values of n and c by substituting $p' = \frac{r-rc}{4}$ into the binomial distribution.

3.3 Choosing parameter values

Assuming that a party only knows the standard deviation of their error process, and wishes to guarantee that they

measure honestly, the agreed value of e will be related to the agreed value of c by:

$$e = \frac{\sigma}{\sqrt{1-c}} \quad (25)$$

Hence if a confidence of $c = 0.99$ is required then a value for e of 10σ is required. This is highly conservative.

If a party understands more details of their error distribution they may be able to accept a tighter bound on e .

A party may negotiate values of e and c such that the true probability p_t of an erroneous value is less than the Chebychev bound. In this case the party will be able to insert purposeful erroneous value in proportion $p - p_t$ to the total size of the log n . Note that the client, through its ability to issue or withhold service requests controls the size of the log.

Given the choice of SLA parameters, this behaviour is impossible to prevent. However, negotiating tighter bounds for e and c will reduce the degree of cheating possible regardless of the true distribution of error in the measurement process of either party. The accuracy of measurement guaranteed may therefore be regarded as a discriminating point in a competitive market of services governed by SLAs.

For example, for a measurement confidence $c = 0.95$ and a type I error rate of $\alpha = 0.05$, in a log of size $n = 1000$ we would tolerate up to $d'_0 = 16$ differences between the party's log and another trusted log with unspecified distribution conforming to the accuracy constraint. If the measurement regime of the untrusted party is in fact perfect with $\text{pr}(|X_i - \mu_i| > e) = 0$, then the party may be confident in introducing up to 16 purposeful errors into its log prior to reconciliation between the parties.

4. RELATED WORK

The theoretical contributions presented in this paper were developed in support of the ongoing development of the language for ASP SLAs, SLAng. SLAng is specifically designed for describing SLAs for scenarios such as that depicted in Figure 1. As a result of the analysis presented in Section 2.3, we incorporated in SLAng the assumption that the client and service provider should always be technically adjacent and therefore observing the same events at the service interface. SLAng incorporates a number of different types of constraints, including timeliness and reliability that are mutually monitorable. The semantics also state that all events referred to by the SLA for the purpose of varying the application of constraints must be monitorable by both parties. Finally, the semantics of SLAng describe the need for reconciliation of evidence between the parties, and incorporates the accuracy constraint developed in Section 3.1.

SLAng is described using precise meta-modelling techniques derived from the MDA toolset [11]. The specification is machine readable, and may be used to derive an editor and consistency checker for the language. The specification and related tools are available as an open-source project for inspection and reuse [2]. A description of the particular design of constraints in the language will be the subject of a future publication.

SLAng competes with a number of current and prior efforts to design a language for ASP SLAs or service offerings, most notable WS-Agreement [6], WSLA [7] and WSOL [13]. To the best of our knowledge, no previous language explicitly addresses the need for monitorability, or provides any

constraints on or discussion of measurement error either in the design of the language or any of its related documentation. However, all of the languages cited here require or permit extension to define metrics, so they have the potential to address these issues.

The management of error in performance measurement for analysis and benchmarking is an important related topic which has been well covered by prior work [8].

Concepts related to monitorability have been touched upon by previous work in the area of policy management. It is conventional in policy languages to define rights and obligations that may attach to managers [12]. These rights and obligations are scoped according to *management domains* containing policy objects, where a manager ‘sees’ one or more management domains. Assuming management domains are correctly modelled, the monitorability of policy objects from the point of view of managers who must execute policy may be ensured at parse time in languages such as Ponder [5].

The execution of an obligation policy is usually the responsibility of the manager to whom it applies. It may be that the manager is not trusted so there is a requirement to check that whether an obligation has been fulfilled. [3] provides an algorithm for determining whether obligations conforming to a given model have been fulfilled, but assumes that all relevant events are visible. [4] provides a logic whereby accountable managers may prove that an obligation has been fulfilled with reference to a log of actions. A notion of observability of actions is introduced to constrain the model. However, accountability relies on a universally trusted logging system being available within the manager’s domain. It is not clear how this could be implemented, or if it could be used to monitor negative obligations. To the best of our knowledge, no work has yet been done on checking policies for monitorability under conservative trust assumptions similar to those adopted in this paper.

A highly influential paper with apparent relation to our work on accuracy is Lamport’s clock synchronisation paper [9]. The accuracy constraint discussed in the preceding sections does not require parties to synchronise clocks in order to measure the timing of mutually observable events. Instead the constraint requires a measurement to be accurate with respect to the true time, and the means by which the parties should achieve this is not relevant.

Lamport described an algorithm with a bounded error for synchronising distributed clocks. This may potentially be helpful in situations such as our example scenario where the parties to the SLA are technically adjacent and so could share synchronisation information, and where the constraints are primarily concerned with relative rather than absolute timings. Measuring the time of events with high global accuracy is difficult, and may require specialist equipment, for example a GPS receiver, so a synchronisation based approach may be preferable. However, it will be necessary to determine how this interacts with trust assumptions we have made.,sec-asp For example, synchronisation protocols should not be employed that allow one party to maliciously alter the clock of another with whom they have an SLA.

5. SUMMARY

This paper has provided three significant contributions to the theory underlying SLAs:

First, we have presented and motivated a model and analy-

sis technique for reasoning about the monitorability of systems of SLAs.

Second, we have instantiated that model to perform an analysis on the important example of client/server computation taking place across a network owned by one or more third parties. In the case that the network is owned by a single provider, and trusted monitoring is not provided using any technical solution (such as tamper-proof monitors), we have demonstrated for latency constraints that the highest level of monitorability possible is for all SLAs to be mutually monitorable, and we have described the single configuration in which this holds, namely that the ISP offers the client an SLA at the client’s interface to the network, and the service provider offers an SLA to the ISP at the service provider’s interface to the network. This is an extremely significant result as it implies that if end-to-end QoS is to be achieved using legalistic SLAs in the Internet then a major change will be required to the business model that ISPs currently operate. Alternatively, if this is not possible, it might suggest the criticality of future research into embedding trusted monitoring solutions into network and service-provision infrastructure.

Finally, we considered support for SLAs that are mutually monitorable but no better. In this case, parties must agree on the behaviour of a service prior to determining the penalties to be paid in relation to a particular SLA. We observed that a naive constraint that the parties report service behaviour honestly would be impractical due to the inevitable presence of error in any measurement, but that without such a constraint an honest party would have no recourse should an agreement fail to be met. We therefore designed a reporting constraint that described a limit on the number of errors a log could contain based on its size, the stated confidence the parties have in the measurements, and an agreed type I error rate for the test. We showed that this constraint could be approximately monitored by using a statistical hypothesis test by comparing the a log of measurement under test to a second log with known error characteristics. Naturally, approximate monitorability fails to detect a degree of cheating, which we quantified.¹

6. REFERENCES

- [1] Implementation of tools for monitorability analysis, 2006. <http://ucslang.sourceforge.net/monitorability.html>.
- [2] The SLAng language and tools, 2006. <http://ucslang.sourceforge.net/>.
- [3] C. Bettini. Obligation monitoring in policy management. In *Third International Workshop on Policies for Distributed Systems and Networks (POLICY’02)*. IEEE Computer Society, 2002.
- [4] J. Cederquist, R. Corin, M. Dekker, S. Etalle, and J. den Hartog. An audit logic for accountability. In *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’05)*. IEEE Computer Society, 2005.
- [5] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *Policy 2001: Workshop on Policies for Distributed Systems and*

¹Thanks to Franco Raimondi for his assistance in developing this material.

- Networks*, number 1995 in Lecture Notes in Computer Science, pages 18 – 39. Springer-Verlag, 2001.
- [6] Global Grid Forum. *Web Services Agreement Specification (WS-Agreement) Version 2005/09*.
 - [7] International Business Machines (IBM), Inc. *Web Service Level Agreement (WSLA) Language Specification*, January 2003.
 - [8] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, Inc., 1991.
 - [9] L. Lamport. Time, clocks, and the ordering of events in a distributed system. In *Communications of the ACM*, volume 21. ACM Press, July 1978.
 - [10] J. Skene and W. Emmerich. Precise service level agreements. In *26th International Conference on Software Engineering (ICSE)*, Edinburgh, UK, May 2004. IEEE Press.
 - [11] J. Skene and W. Emmerich. Specifications, not meta-models. In *ICSE 2006 Workshop on Global integrated Model Management (GaMMa 2006)*, pages 47 – 54. ACM Press, June 2006.
 - [12] M. Sloman. Policy driven management for distributed systems. In *Journal of Network and Systems Management*, volume 2. Plenum Press, 1994.
 - [13] V. Tosic. *Service Offerings for XML Web Services and Their Management Applications*. PhD thesis, Ottawa-Carleton Institute for Electrical and Computer Engineering, 2002.
 - [14] Wikipedia. *Chebyshev's Inequality*, 2006. http://en.wikipedia.org/wiki/Chebyshev%27s_inequality.