The BabelNEG System - A prototype Infrastructure for protocol-generic SLA Negotiations

Sebastian Hudert
Department of Information Systems Management, University of Bayreuth
Universitaetsstr. 30
95447 Bayreuth, Germany
sebastian.hudert@uni-bayreuth.de

Torsten Eymann
Department of Information Systems Management, University of Bayreuth
Universitaetsstr. 30
95447 Bayreuth, Germany
torsten.eymann@uni-bayreuth.de

ABSTRACT

Visions of the next-generation Internet of Services are driven by digital resources traded on a global scope. For the resulting economic setting, automated on-line techniques for handling services and resources themselves, for advertising and discovering as well as for the on-the-fly negotiation of proper terms for their use are needed. Hence, a flexible infrastructure for the respective management of services and associated service level agreements is mandatory.

In this paper we present a novel approach for such an infrastructure, building on software-agent technology and an expressive but still machine manageable protocol description language, capable of specifying a multitude of different negotiation protocols. It supports the discovery of services with appropriate SLA negotiation styles as well as the actual SLA negotiations based on the chosen protocol description documents.

Keywords

service level agreements, internet of services, electronic negotiations, software agents, quality of service

1. INTRODUCTION AND MOTIVATION

Current developments in the area of Information Systems show a tendency towards massively distributed infrastructures, consisting of highly specialized digital resources. Today's Internet of mainly human interactions will evolve towards a socio-technical and global information infrastructure, where humans as well as software agents, acting on their behalf, continuously interact to exchange data and computational resources. This vision can be observed in both research and industry alike and is commonly referred to as the Internet of Services (IoS) [27, 25].

Building on currently applied computing paradigms, such as Service-oriented [11], Grid [12] or Cloud Computing [6], the IoS vision defines highly dynamic networks of composable services, offered and consumed on demand and on a global scope. Taking such ideas one step further, it rigorously focuses on the goal of an Internet-based service economy, similar to the real-world service sector. Digital services will be offered over electronic service markets, purchased by respective customers and then combined with internal or other external services to business workflows of varying complexity.

Hence, the IoS will primarily focus on new business models and the commercial application of distributed computing, concerning trading processes down to the level of an individual service, and the subsequent charging based on its usage and delivered quality-of-service (QoS). In such a setting even very small and specialized companies can find a niche in the digital economy where they can compete with the ubiquitous international enterprises, which in turn have to face a much higher competition on the global market [9].

Summarizing the IoS scenario thus results in the following characteristics [27]:

- The IoS focuses on a (potentially huge) set of electronic services of varying complexity.
- These services will be employed in potentially mission-critical business processes and thus have to fulfill a (pre-negotiated) set of QoS guarantees as stated in a Service Level Agreement (SLA)[1].
- New business models will cope with the possibility of trading even very fine-granular services and charging them based on their actual usage.
- It will consist of a global set of SPs and SCs, negotiating over digital services as well as some mediating nodes, such as service brokers.

Two of the main challenges for the IoS, from a commercial perspective, are reliability of the services traded and the technical infrastructure underlying the service economy.

[1]A SLA is a structured document, describing a bilateral agreement between a service provider (SP) and a service consumer (SC) on the terms and conditions of the invocation(s) of an (electronic) service.

10^{th} International Conference on Wirtschaftsinformatik,
$16^{th} - 18^{th}$ February 2011, Zurich, Switzerland

The need for guaranteed reliability and service quality becomes more prominent, as no longer the question of "who provides the service?" matters but only whether he is able to achieve the requested result.

Since such scenarios inherently lack the applicability of centralized QoS management, service guarantees must be obtained in the form of bi-lateral SLAs assuring service quality across individual sites [18]. These SLAs subsequently act as a signed contract governing the actual service invocation [6], enabling the structured monitoring and assessment of the service's compliance[2].

A very crucial part of the SLA-based service life cycle can be seen in the discovery and, above all, the negotiation phase. All subsequent steps (binding, execution and monitoring, post-processing etc.) depend on the SLA documents which were agreed-upon in this phase. On that account, we focus on a the negotiation and prior discovery of SLAs for our work presented in this paper.

Aiming at the negotiation phase, economic reserach claims that differences in system configuration, or the services actually traded, demand different negotiation protocols in order to reach the highest-possible efficiency of the overall system (see for example [19]). Based on these findings and the global context of the envisioned scenario it is not likely, or even efficient, that only one central marketplace for electronic services will emerge, offering a single, known protocol. Instead a system of marketplaces offering different protocols will probably emerge, each of which is best be suited for a given context.

Fortifying this, we argue that restricting SCs in that they are only able to interact with one distinct service market they were implemented for (and are therefore only technically compatible with the applied negotiation protocol), unnecessarily decreases the potential flexibility and efficiency of the whole system. SCs should be able to buy, and therefore negotiate about, any fitting service, regardless of the market it is offered in, and thus regardless of the protocol with which it is offered. SPs, on the other hand, should be able to offer their services with the protocol best fitting to the current market situation instead of being restricted to a given protocol by the market infrastructure.

Also, given the dynamic nature of distributed workflow executions and the increased complexity of global service selection manual negotiations of the human users are by far not efficient enough. This process should be automated by electronic software agents that negotiate on the users' behalf [17].

The research goal of our work is thus to develop a service-oriented infrastructure supporting software agents to discover and negotiate about electronic SLAs and not restricting them to a pre-defined negotiation protocol.

[2]In our work we focus on electronic SLAs as needed in the anticipated fully automated setting. Such documents are mainly used for resource management and scheduling in various research projects [22] and mark a promising approach for representing real-world, legally binding, contracts in an electronic way.

The remainder of this paper is structured as follows: In section 2 we will present a short overview on related research projects in the area of electronic SLA management systems and the IoS. Next, the main contribution of this paper is presented in section 3, the design of our SLA negotiation infrastructure. Section 4 will provide information on the developed proof-of-concept prototype system and the evaluation steps already conducted. Finally, we conclude our paper with a short summary and the identification of future work.

2. RELATED WORK

A significant amount of research projects exist, dealing with distributed QoS management based on SLAs. Such efforts have risen after traditional distributed systems came to maturity and reliability came into focus. Some of the most notable approaches were presented by Ludwig et al. [18], Yarmolenko and Sakellariou [26] or Tosic et al. [30]. These works mainly address the internal structure of SLAs, the relevant service metrics and SLA-based resource management and scheduling mechanisms.

Building on such theoretical works, an ever growing amount of research projects, such as CoreGRID[3] or SLA@SOI[4] employ SLAs for resource management and designed respective SLA management systems. Also the ongoing Web Services Agreement [1] standardization effort at the Open Grid Forum[5] shows the growing interest in SLA-based QoS management from both research and industry.

On the other hand a variety of different negotiation settings and respective protocols for both electronic and real-world markets have been introduced (see for example [24] for very fundamental work on negotiations). As a next step, these findings were ported to the digital world, forming the new research discipline of Electronic Negotiations [2]. This lead to the definition of formal descriptions and characterizations of given negotiation protocols [28] as well as first attempts to software infrastructures for (electronic) negotiations (see for example [32]).

After software agent technologies [31] reached maturity it was only a logical next step to employ the mechanisms developed therein for the implementation of electronic markets [23].

Surprisingly, there is little research done in combining the economic considerations on the one hand and the technical developments on the other hand. No more than a couple of research groups address the agent-based, electronic negotiation of SLAs; resulting infrastructures were presented for example in [7] or [21]. Even those projects mainly focus on static and centralized architectures within which only one particular, and fixed, negotiation protocol is implemented. Hence, they allow for the definition of individual service markets, but still lack the possibility for SCs to migrate from one market, and thus one negotiation protocol, to another (in analogy to a real-world economy).

[3]http://www.coregrid.net
[4]http://sla-at-soi.eu
[5]http://www.ogf.org

Although the need for protocol-generic systems is widely agreed-upon (see for example [19]), only a few projects incorporate the mere possibility of different protocols within one infrastructure; [17] and [3] being two of the most prominent examples. However, both still lack important flexibility by restricting the negotiation protocols to a small and fixed set and by building on static, centralized architectures without appropriate discovery mechanisms.

3. INFRASTRUCTURE DESIGN

Before describing the actual system design, we will now present the results of a literature-based requirements analysis, underlining the need for identified protocol-generic SLA negotiation systems.

3.1 Requirements Analysis

Among the abovementioned researchers, a common agreement exists on which requirements are posed on electronic SLA management infrastructures for the IoS. The most significant requirements concerning the discovery and negotiation phases include:

R1 After the discovery phase all parties must have a common understanding of the protocol to be executed in the negotiation phase [17].

R2 This common understanding must be generated dynamically at runtime [4].

R3 Services (and thus SLAs) of different complexity must be negotiable [19].

R4 Different marketplaces and protocols (fixed-price catalogues, bargaining, auctions etc.) even within one infrastructure are needed for different services to be traded [19]. In order to cope with the highly dynamic IoS environment the available set of protocols should not be restricted a priori [15].

R5 Software agents should act as negotiators [17].

R6 Intermediaries, such as auctioneers or brokers, should be present [6].

Although most of these requirements were even identified in the context of either one of the abovementioned research projects, no infrastructure to date fulfills all of them. Especially the claimed protocol-flexibility of SLA negotiations, in particular at run time, is not reached or even not addressed at all. In the following we will present the design of our system, aiming at closing that gap.

3.2 Abstract Design Idea

The basic design idea underlying this work is to offer a given product (SLA for an electronic service) independently from the way an agreement concerning this product can be attained (negotiation protocol). This allows for flexible combinations of product and protocol to be chosen for each market situation individually.

Such an approach has many analogies in real-world settings. For example a TV set, offered at an electronic retailer implicitly states that the only way to negotiate about it is to accept the stated price. This protocol thus corresponds to a classic cataloque pricing model. Then again, the same TV set, sold on an online auction platform such as Ebay[6], implies that the consumers have to outbid each other until a certain deadline occurs. This in turn corresponds to an English Auction protocol. Although the product sold in both cases is exactly the same (a new TV set of a given type) the negotiation protocols applied are quite different. Transferred into the electonic SLA management scenario, this means that a given SLA (template[7]) under negotiation could be offered with a whole set of different negotiation protocols at different points in time, best fitting the current market situation.

Since software agents will be employed for the service management within our systen (R5), the negotiation protocol applied for a given service is not only decoupled from the actual SLA but must also be made explicit in terms of its communication rules. This allows for run time adaption of the SC agents to the respective protocol.

To this end, the designed infrastructure will build on a conceptual architecture of machine-readable description documents, as described in the following.

3.3 Service Description Documents

As just introduced, a set of service description documents is needed enabling a) the discovery of an appropriate service or respective SLA (template) and b) the description of the negotiation protocol used to reach an agreement. For that purpose three different data structures were designed:

- Service Type (ST): definition of the functional and non-functional aspects, a given class of services can offer.
- Extended SLA Template (EST): definition of initial QoS guarantees (building on the non-functional aspects given in the respective ST) as an input for the subsequent negotiation as well as the applied negotiation protocol.
- Service Identificator (SI): identification of an individual service instance along with links to the associated ST and EST documents.

Figure 1 gives a short overview on these documents and their relations.

[6]http://www.ebay.com/

[7]Most of the present SLA (management) systems incorporate the possibility to define SLA templates (see for example [1]). A SLA template is basically a partially filled out SLA document, that is offered from the SP to potential cunsumers. It regularly marks the starting point for an actual negotiation process, during which the already stated service guarantees (also called service level objectives (SLOs)) are altered according to the preferences of the negotiators until the final SLA is reached. The internal structure of a SLA template is equivalent to an actual SLA document. The only difference is, that a set of rules is optionally defined, helping the SC to identify (combinations of) SLOs that are a) valid and/or b) acceptable by the SP (see for example [1, pp. 30-33]).

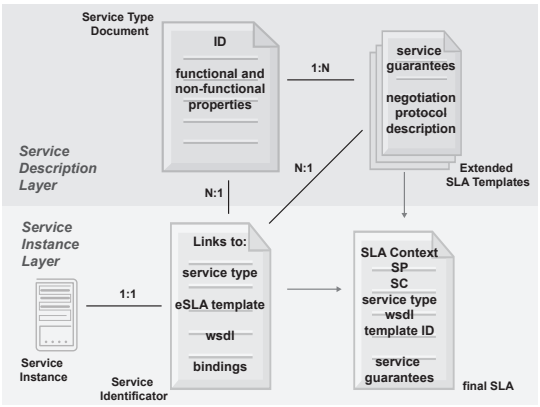

Figure 1: Document-based Architecture

Note: All data structures were defined on the basis of XML Schema [29] due to the massive adoption of XML [5] for communications in service-oriented systems.

3.3.1 Service Type

The basic goal of the ST document is to describe a class of service instances in terms of their functional capabilities and the non-functional parameters, such as throughput or price.

The functional aspects are primarily used to search for a service, taking into account the search criteria as received from the user / service requestor. In contrast, the non-functional parameters, or rather their actual values, are mostly the very object of any SLA negotiation. These parameters' actual values vary during run-time, depending on the current workload. The incorporation of respective monitoring and enforcement components at design-time however, offers the management agents the possibility to negotiate about such metrics and subsequently enforce the resulting guarantee, even at run-time.

3.3.2 Extended SLA Template

In our work, the SLA template concept was not only also used to define initial SLOs (R3), but additionally extended to also include a description of the applied negotiation protocol[8].

The respective part of the EST is created, building on a negotiation protocol description language developed by the authors (a previous version of which can be found in [14]). It builds on simple parameters where possible and, if more complex restrictions are needed, allows the application of any desired (external) rule language within some of the language elements.

A respective protocol description provides the querying SC agent with information about the:

- *negotiation context* (permitted agents and their permissions and obligations),
- *object* (actually negotiable SLOs),
- *offer restrictions* (constraints posed on the negotiable SLOs, such as upper or lower bounds of acceptable values),
- *allocation* (matching function applied for winner determination),
- *information policy* (information access rules, as needed for example to define sealed-bid or outcry negotiations)
- *negotiation process* (definition of the allowed behavior of the involved agents, employing the event-condition-action paradigm).

3.3.3 Service Identificator

Each individual service instance is finally defined within a respective SI. Such a document describes where exactly this service can be found (important for the actual binding process), what its type is and which EST is offered for it. It thus consists of the following elements:

- *serviceID*: identifier for this particular SI. This element should be of the type URI in order to ensure uniqueness in a global setting.
- *serviceTypeID*: link to the description of this service's type
- *slaTemplateID*: link to the EST offered for this service
- *wsdlFile*: reference to the WSDL [8] file, describing the actual service interface in terms of operations with input and output parameters as well as error types.
- *negotiationCoordinator (NC)* and *serviceProvider*: these two elements represent role bindings for this service[9].

3.4 Protocol Design

In order to support both, the discovery and the negotiation of SLAs, a set of simple protocol primitives have been developed, building on the abovementioned data structures.

3.4.1 Discovery Phase

The discovery phase basically represents the set of activities ultimately leading to a situation where potential business transaction partners (SCs and SPs) know one another and can start a negotiation process. This means the discovery phase is supposed to support a given SC to find one or more SI documents fitting the search criteria it received from the user (R1 and R2). To this end, the SPs should be able to publish the service it offers (in terms of the respective description documents), in a way that it can be found by potentially interested SCs.

For our proof-of-concept implementation, a very simple registry node, which can be found via a broadcast-based discovery protocol, was chosen (see figure 2 for more details). Such a simple architecture was considered sufficient for investigating the research question at hand. Future versions

[8]Since this aspect is very closely linked to the SLOs already stated in the template it was considered appropriate to integrate it into the same data structure.

[9]For more details on the role architecture, see subsection 3.5.

however, will probably favor more robust mechanisms, such as Peer-to-Peer discovery approaches.

For either the registry discovery, the publication of one or more service documents or the discovery of those a set of message types was defined. These messages strictly follow the internal structure of SOAP [13] messages to allow a seamless integration into present Web Service infrastructures.

At first the SP, upon receiving a request to sell a given service, publishes the respective service description documents at the registry node. Subsequently it waits for a SC to request admission to the actual negotiation. The SC process is on the other hand triggered by the reception of a user request for a particular service. In a first step, the SC requests all SIs fitting the search criteria and retrieves the (until then unknown) ST and EST documents from the registry. Given a list of adequate SIs were found, the SC chooses one of them and tries to start / join a negotiation via an explicit admission step at the NC agent (for each negotiation protocol exactly one agent adopts this role; see subsection 3.5 for more details). In case of a successful admission the SC and SP now engage in the actual negotiation process.

3.4.2 Negotiation Phase

Since the main goal of our work is to define an infrastructure for protocol-generic negotiations, no single negotiation protocol can be identified for this phase[10]. Rather, a set of negotiation message types along with their respective contents was defined. These messages can then be used in a given negotiation process, orchestrated according to the protocol description in the EST document (R1, R2 and R4).

After a thorough literature review on currently applied negotiation protocols (as listed for example in online libraries such as [10]), a minimally necessary set of such messages could be identified, consisting of *offer(toSell)*, *accept*, *reject*, *callForBids* and *stillInterested* messages. Using these message types a variety of different protocols can be described in an EST document and subsequently processed by the service management agents[11].

3.5 Role-based Architecture

The developed prototype infrastructure builds on a defined set of roles, the service management agents can adopt (see figure 3):

The two basic roles present in this system are the SC and the SP, representing buyer and seller agents.

Additionally a set of registry services / agents (RA) are needed for supporting the publication and discovery of service description documents. Finally, the NC role represents the agents mediating negotiation processes as a broker agent (R6).

[10]Therefore no distinct process model for this phase (analogue to figure 2) can be given. Individual Negotiation phases always differ, depending on the applied protocols.

[11]A set of mutually very different negotiation protocols, designed only building on these message types, can be found at `https://sourceforge.net/apps/mediawiki/simis/index.php?title=BabelNeg`

Both SP and SC agents mainly offer an interface to one another, allowing them to send and receive messages related to the discovery and negotiation phases. Additionally, each of theses roles offers one routine to external users of the infrastructure: A SP agent offers a method for publishing and selling and a SC one for discovering and purchasing a service on a user's behalf, respectively.

A RA only accepts discovery related messages as it does not take part in any other phases of the life cycle. Finally, NC agents are responsible for admission of SCs or SPs to and potentially mediation of a given negotiation. Hence, they again offer methods for exchanging respective messages.

In the following the internal routines of each role, as implemented in the proof-of-concept prototype, will be further detailed.

3.5.1 Service Provider

A SP agent's basic purpose is to publish a given service to potential customers and sell it to them subsequently. On an abstract level a SP agent first receives the request for publishing a service and the respective description documents (triggering a state change from IDLE to PUBLICATION). Now it stores all respective documents at a RA node (which potentially has to be discovered first). If successful, it transits to the BUSY state, in which the service is continuously offered, negotiated about and executed. The SP only exits this state when the service is taken off-line or is re-deployed.

3.5.2 Service Consumer

The default process for a SC is to receive a service request and move to the discovery phase, during which a set of fitting SI documents are retrieved. After choosing an appropriate SI the SC agent requests admission to the respective negotiation, joins the negotiation process if admission was successful and finally invokes the service in the EXECUTION state if it could win the negotiation.

A fundamental principle was implemented all along the process of a SC agent: the reluctant escalation backtracking in terms of internal states. In case of a failed attempt to move to the next state (e.g. from the DISCOVERY to the INITIATION_NEGOTIATION state) it checks its options for another such attempt. In this case it checks whether or not any other SI was found, for which it could request admission to the respective negotiation. Only if no such options are available it moves back to the predecessor state and checks whether it can start over from there or if even there no other options are available and so on. This principle ensures the agent to fully exploit all possibilities it could discover for reaching an agreement.

3.5.3 Protocol-generic SC Strategy Component

During the actual negotiation phase a protocol-generic strategy component is employed by the SC agents. All routines needed for adaption to a new negotiation protocol are implemented herein.

When admitted to a negotiation process, the SC instantiates such a generic negotiator (GN) component and passes over the ST, EST and SI documents, as applied for the respective

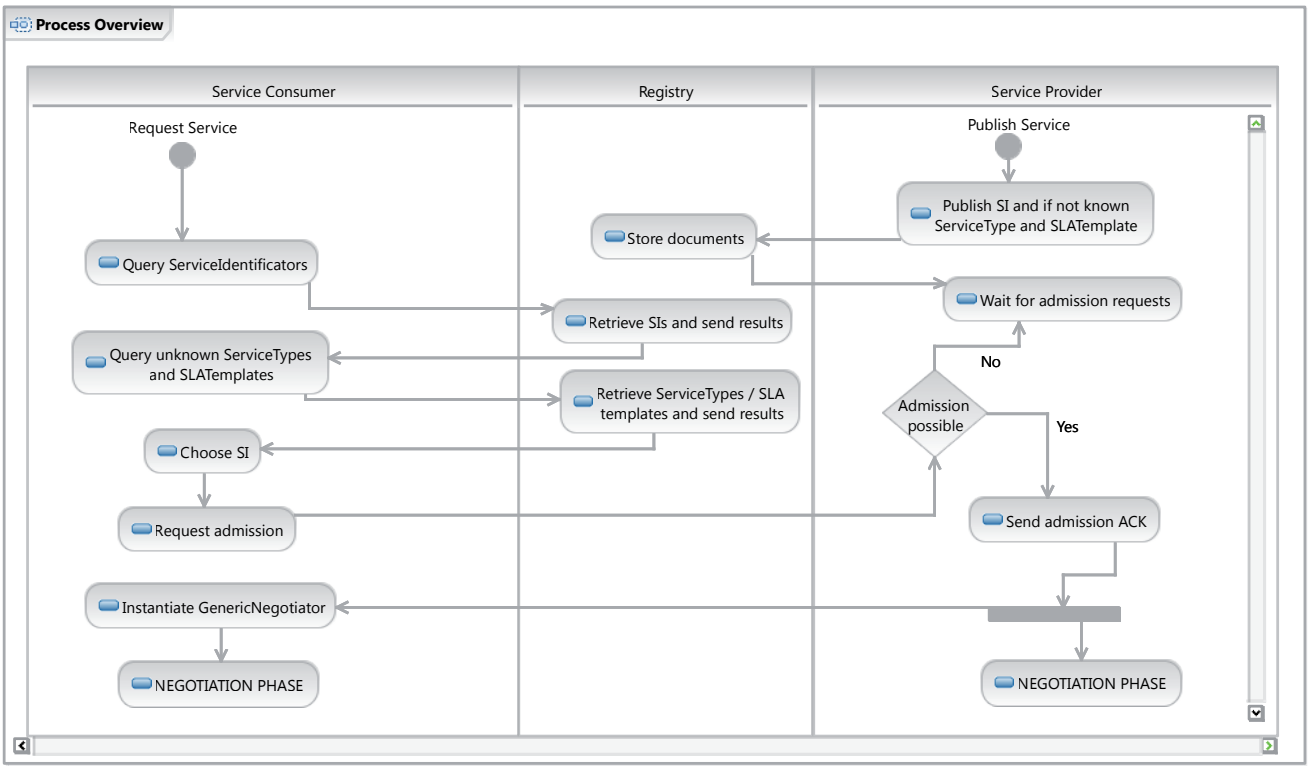

Figure 2: Discovery Phase Sketch

negotiation, as well as all internal constraints received from the user (such as reservation values for certain SLOs).

Once all this information was extracted the GN checks whether it has to start the negotiation (pro-active protocol). If so, two possible actions can be defined: this agent can be allowed to post an offer or to simply accept all the the values stated in the EST (corresponding to a catalogue pricing model).

Here a basic principle of the GN becomes obvious: it always seeks to maximize its utility and thus will always choose some actions over others (if both are allowed). In this case it will first check, whether an offer is possible (such checks are always done by inspecting the respective event-condition-action tuples stated in the EST), which could further improve the currently offered agreement. If so, an offer will be sent; if not it will simply accept the current values, as no negotiation on them is allowed (if they are acceptable).

The same principle applies throughout the whole negotiation process. Whenever a negotiation message is dispatched to the GN it checks its options: In case of a reject or accept message the negotiation is over. It simply processes the result in that it passes the respective information / reached agreement to the SC agent. In case of a callForBids message it creates an offer and sends it to the SP agent[12]. All these possibilities are straightforward, as they don't give the GN any option to choose among a set of possible actions.

When receiving an offer, this could potentially change. If the offer is not completely rejectable (this is checked by iterating over all involved slo constraints) and a counter offer is possible, the GN will always do so. A counter offer is always the best option, as it potentially increases value of the agreement for the user. If this is not possible, the GN will check whether a stillInterested-message is allowed, providing it with the possibility of an ongoing negotiation, even if it cannot actively influence the changing of the negotiated values. If even this is not possible it will finally check whether the received offer can be accepted or in the end rejected completely and do so. This routine gives the GN the possibility to react on incoming messages in a way that maximizes its further options during the negotiation and in the end potentially its utility in terms of the reached agreement.

3.5.4 Registry

The RA node is basically just a placeholder element for any discovery mechanism used in future versions. It was designed to simply receive register messages and store the corresponding data into an internal data structure. This is for example used when a SP publishes a new SI document. Supporting the discovery of service documents it also offers the possibility to query the stored SI, EST and ST documents. No complex internal states are maintained and changed throughout its life time. It simply processes requests to register or query service documents or answers registry discovery messages.

[12]Offer messages are simply created by iterating over all not yet fulfilled user constraints and creating a counter offer value for each. These values are then combined to one offer message.

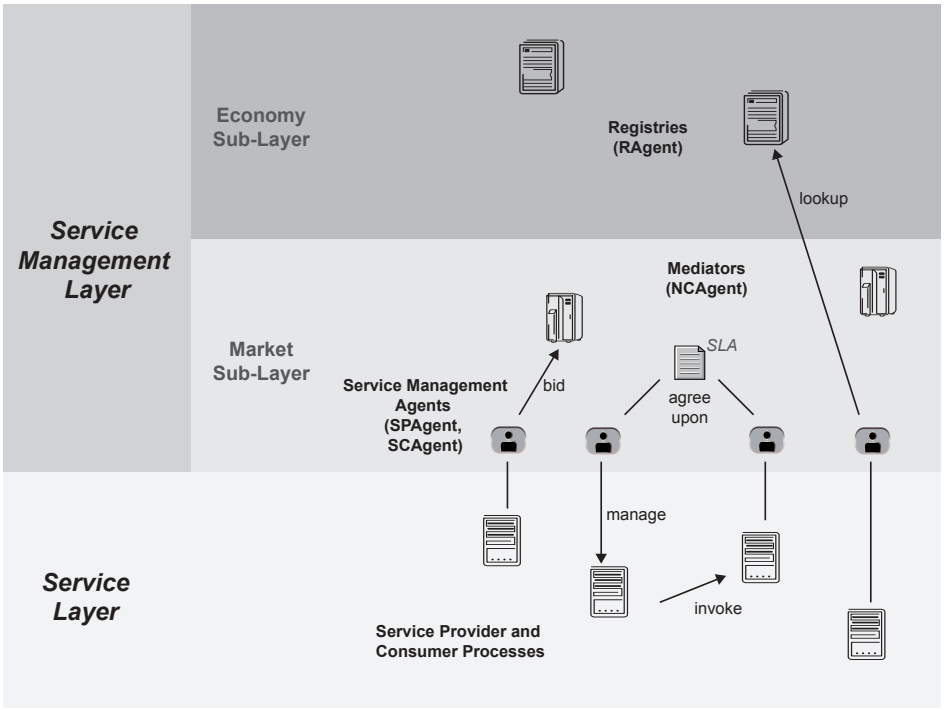

Figure 3: System Architecture

3.5.5 Negotiation Coordinator

The main task of a NC agent is to handle the admission of SCs to a given negotiation. For this it offers the possibility to submit respective messages along with a set of credentials, as needed for the admission decision. Additionally, if the negotiation phase is also assigned to the NC (mediated negotiation), it must also be able to process incoming negotiation messages. Just as described for the SP and SC agents, these are simply forwarded to a negotiation strategy component, which in turn provides the protocol-specific functionality.

Similarly to the RA nodes, a NC thus does not expose complex internal states and state changes. Admission requests are evaluated and answered based on the received data and the implicit service availability information, and negotiation related messages are simply forwarded to the strategy component.

4. PROTOTYPE IMPLEMENTATION AND EVALUATION

The presented mechanisms were implemented in a Java-based proof-of-concept prototype infrastructure, building on the agent-based IoS simulation toolkit SimIS[13]. The simulation experiments, presented in the following, represent one evaluation step for our work, allowing us to model different environmental settings (i.e. market configurations) and demonstrate the feasibility of our approach therein. In a second step, future versions of the developed components will be ported to a productive IoS platform dealing with the whole service life cycle.

[13]http://sourceforge.net/projects/simis

SimIS was co-developed by the authors and aims at providing researchers with a comprehensive framework for investigating distributed algorithms or protocols within the context of the IoS vision. Building on the generic Recursive Porous Agent Simulation Toolkit (REPAST) [20], it proposes a two-tiered architecture dividing the overall system into an *Application Layer* and an *Infrastructure Layer* (see figure 4).

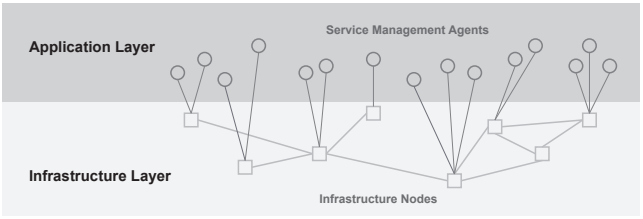

Figure 4: SimIS Toolkit

The Infrastructure Layer models topological settings of the IoS. The basic idea is that all Application Layer Agents / Services are linked to a single Infrastructure Agent each, which is representing their server platform, mainly performing message handling tasks.

Within the Application Layer the actual services of the IoS vision are modelled, communicating via the offered message objects and routing functionality. Each service, i.e. Application Layer Agent, is to be implemented as a plain Java class and can therefore exploit the full potential this programming language offers. A more detailed description of the SimIS toolkit can be found in [16].

The agent types developed for our infrastructure prototype were consequently implemented as specific Application Layer agents within SimIS. Each of the management agents (SCs and SPs) is accompanied by a strategy object, the SCs with the protocol-generic negotiator and the SPs with a strategy distinctly fitting to the offered protocol (as parameterized during startup).

In a first step the document-based architecture and the respective discovery mechanism were tested by running a variety of different configurations within the simulated IoS setting (varying numbers of SCs and respective service requests, SPs with a given service / protocol combination, and RAs supporting the discovery process). Both the protocol steps applied and the RA implementation were able to prove effectiveness very well.

Secondly an initial set of negotiation protocols was chosen to be a) described with the developed language and b) applied within SimIS to show whether or not the GN is able to correctly extract the relevant information and take part in the subsequent negotiation.

For the subsequent evaluation runs the Alternate Offers, an English Auction and Double Auction Protocol were instantiated. This selection covers 1:1 (Alternate Offers), 1:N (English Auction) and M:N (Double Auction) settings and can thus be regarded as a representative subset of the most commonly used negotiation protocols [14].

In the Alternate Offers Protocol, both parties basically exchange offer messages. Upon receving such a message the agent has three options: accepting the offered values, rejecting them ultimately or creating a counter-offer message. In the English Auction, the SP offers a given value for the negotiated metric (in the shown experiments, the price) to all SCs, which in turn can answer with stillInterested messages, given they are still interested in the negotiated service under the offered conditions. In the next round the SP increases the value of the negotiated SLO and offers it to all remaining SC in the same way. This process is repeated until only one SC is left that accepts the offered value. During a Double Auction both, SP and SC, post messages indicating their service offers or demands. The broker (NC) subsequently matches these according to a defined matching function and passes the results to back to the SPs and SCs.

For demonstration purposes, we configured the simulator with ten infrastructure nodes, upon which 40 SPs and 50 SCs are placed. Each SP offers the same service, however 8 of those with the Alternate Offers, 12 with the English Auction and the remaining 20 with the Double Auction protocol, as described above. Additionally one registry node for storing and retrieving the service and protocol description documents as well as one broker for the Double Auction are present.

The SCs generate service requests at random points in time (mean duration between two request arrivals: 50 ticks), starting at tick 60; this way start-up effects of the simulation can be avoided. Each of these agents starts with the same valuation for the offered service but adopts its valuation over time (in case of a successful negotiation it decreases its valuation by 0,05, in case of an unsuccessful negotiation vice versa). Additionally, a set of timeouts were employed for timing of new auction rounds or in case of not answering opponents. The experiments were delimited to a maximum tick count of 20000.

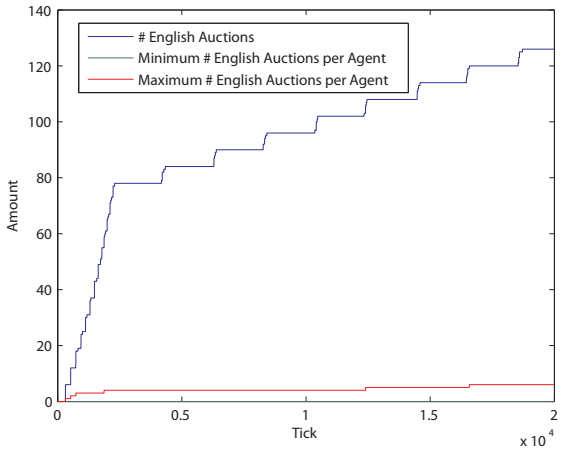

Figure 5: Successful English Auctions

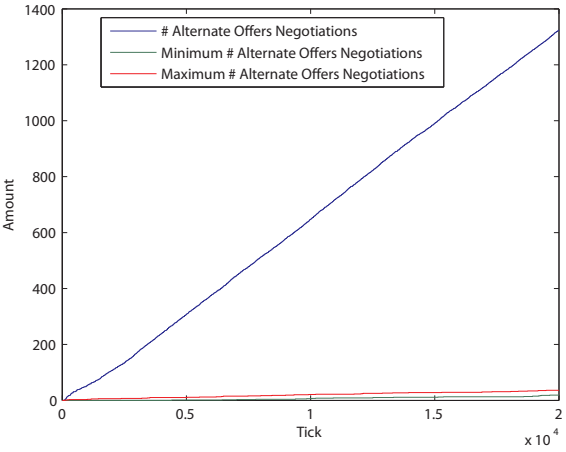

Figure 6: Successful Alternate Offers Negotiations

The main statement to be proven with these experiments is that the GN node is at all able to adapt to different protocols, only based on their description in the EST documents (proof-of-concept). Given the results shown in figures 6, 5 and 7, this assertion can be approved. Each of the introduced protocols was successfully integrated in the actual market behaviour; each protocol type present was executed by a significant number of agents.

This fact is underlined by figure 8 which shows that each

[14]The used protocols, described in terms of UML Sequence Diagrams as well as a respective EST documents, can be found at: `https://sourceforge.net/apps/mediawiki/simis/index.php?title=BabelNeg`

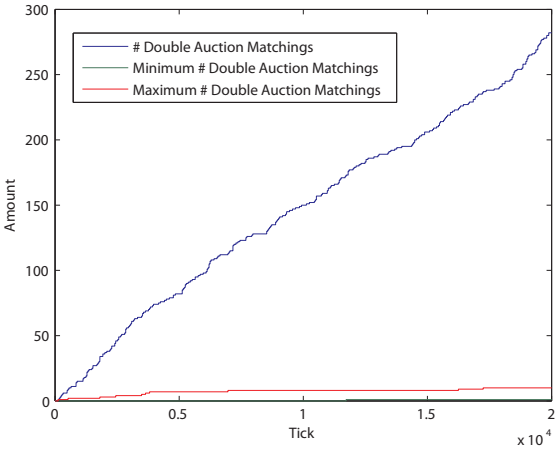

Figure 7: Successful Double Auctions

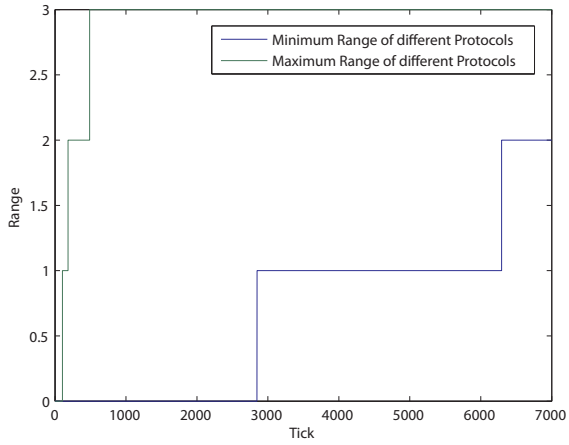

Figure 8: Range of used Protocols

agent could successfully take part in at least two different negotiation protocol types. On the other hand, there exist SC agents that have successfully taken part in all available protocols.

We will now shortly relate the developed system to the requirements stated in section 3.1.

R1 and R2: Dynamically generated understanding of negotiation protocol, during discovery phase

Each SC enters the discovery phase with no prior knowledge about the protocol that is executed in the subsequent negotiation phase. All information needed for a successfull participation in a respective negotiation is coded in the service description documents, which are queried during discovery. Therefore the requirements R1 and R2 are fulfilled by our system.

R3: Support for SLAs of different complexity

The ST, EST and SLA documents provide a very generic service (SLA) description structure. By offering free-form text elements along with the pre-defined and typed elements for quantitatively measurable service aspects, a comprehensive service description can be created. The defined document structures also enable the usage of external, standard languages for describing service characteristics (for example WSDL when describing the service interface).

R4: Support for different negotiation protocols

By employing a structured protocols description language in the EST document, each service (and SLA template respectively) can be offered over a different protocol. The GN strategy component, being able to adapt to the described protocols, supports all protocols that can be described in that language, thus fulfilling R4.

R5: Software agents acting as negotiators

The system architecture, as implemented in the proof-of-concept prototype, heavily builds on sofware agents as instantiations of a particular role. Such components are the basic actors in our system.

R6: Need for intermediaries

Our system incorporates an explicit intermediary role, used for the definition of market brokers, the NC. Therefore R6 is fulfilled by our infrastructure proposal.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a novel infrastructure for the discovery and protocol-generic negotiation of SLAs. For this we employed a set of structured document types and software-agent technology for managing the incoming requests to buy and sell electronic services and negotiate about SLAs respectively. We evaluated our system subsequently, with a set of different negotiation protocols and system configurations, within the context of an IoS simulation toolkit.

Future work will comprise the simulative investigation of more complex scenarios in which a multitude of different brokers and other negotiation protocols are present at the same time. Such evaluation runs will especially focus on the utility gain a protocol-generic infrastructure can provide over one offering only one single protocol. Also, a more thorough investigation of negotiation strategies, able to efficiently cope with different protocols, has to be done.

6. REFERENCES

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification, version 03/2007. 2007.

[2] M. Bichler, G. Kersten, and S. Strecker. Towards a structured design of electronic negotiations. *Group Decision and Negotiation*, 12(4):311–335, 2003.

[3] I. Brandic, S. Venugopal, M. Mattess, and R. Buyya. Towards a meta-negotiation architecture for sla-aware grid services. In *Workshop on Service-Oriented*

Engineering and Optimizations 2008. In conjunction with International Conference on High Performance Computing 2008 (HiPC2008), Bangalore, India, December 17 - 20, 2008.

[4] I. Brandic, S. Venugopal, M. Mattess, and R. Buyya. Towards a meta-negotiation architecture for sla-aware grid services. Techreport, University of Melbourne, August 2008.

[5] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0 (fourth edition). *W3C*, August 2006.

[6] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[7] M. B. Chhetri, J. Lin, S. Goh, J. Y. Zhang, R. Kowalczyk, and J. Yan. A coordinated architecture for the agent-based service level agreement negotiation of web service composition. In *ASWEC '06: Proceedings of the Australian Software Engineering Conference (ASWEC'06)*, pages 90–99, Washington, DC, USA, 2006. IEEE Computer Society.

[8] R. Chinnici, J.-J. Moreau, A. Ryma, and S. Weerawarana. Web services description language (wsdl) version 2.0 part 1: Core language. *W3C*, March 2006.

[9] T. P. Consortium. Texo: Business webs im internet der dienste (german). http://theseus-programm.de/anwendungsszenarien/texo/default.aspx, 2009. last checked: 2010.01.13.

[10] T. F. for Intelligent Physical Agents. |"www.fipa.org. last checked: 17. 07. 08.

[11] I. Foster. Service-oriented science. *Science*, 308(5723):814–817, 2005.

[12] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15:2001, 2001.

[13] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. Soap version 1.2 part 1: Messaging framework. *W3C*, June 2003.

[14] S. Hudert, T. Eymann, H. Ludwig, and G. Wirtz. A negotiation protocol description language for automated service level agreement negotiations. In *Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC 09), Vienna, Austria*, 2009.

[15] S. Hudert, H. Ludwig, and G. Wirtz. Negotiating slas - an approach for a generic negotiation framework for ws-agreement. *Journal of Grid Computing*, 7(2):225–246, June 2009. ISSN: 1570-7873 (Print) 1572-9814 (Online).

[16] S. König, S. Hudert, and T. Eymann. Socio-economic mechanisms to coordinate the internet of services Ű the simulation environment simis. *Journal of Artificial Societies and Social Simulation (JASSS)*, 13(2), 2010.

[17] A. Ludwig, P. Braun, R. Kowalczyk, and B. Franczyk. A framework for automated negotiation of service level agreements in services grids. In *Lecture Notes in Computer Science, Proceedings of the Workshop on Web Service Choreography and Orchestration for Business Process Management, 2006*, volume 3812/2006, 2006.

[18] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck. A service level agreement language for dynamic electronic services. *Journal of Electronic Commerce Research*, 3:43–59, 2003.

[19] D. Neumann, J. Stoesser, C. Weinhardt, and J. Nimis. A framework for commercial grids - economic and technical challenges. *Journal of Grid Computing*, 6(3):325–347, September 2008. ISSN: 1570-7873.

[20] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, 2006.

[21] M. A. Oey, R. J. Timmer, D. G. A. Mobach, B. J. Overeinder, and F. M. T. Brazier. Ws-agreement based resource negotiation in agentscape. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–2, New York, NY, USA, 2007. ACM.

[22] M. Parkin, R. M. Badia, and J. Martrat. A comparison of sla use in six of the european commissions fp6 projects. Technical Report TR-0129, CoreGRID, 2008.

[23] S. Paurobally, V. Tamma, and M. Wooldridge. A framework for web service negotiation. *ACM Trans. Auton. Adapt. Syst.*, 2(4):14, 2007.

[24] H. Raiffa. *The Art and Science of Negotiation*. Harvard University Press, Cambridge, Mass., 1982.

[25] R. Ruggaber. Internet of services sap research vision. In *WETICE '07: Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, page 3, Washington, DC, USA, 2007. IEEE Computer Society.

[26] R. Sakellariou and V. Yarmolenko. On the flexibility of ws-agreement for job submission. In *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*, pages 1–6, New York, NY, USA, 2005. ACM.

[27] C. Schroth and T. Janner. Web 2.0 and SOA: Converging concepts enabling the internet of services. *IT Professional*, 9(3):36–41, 2007.

[28] M. Stroebel and C. Weinhardt. The Montreal Taxonomy for electronic negotiations. *Journal of Group Decision and Negotiation*, 12:143–164, 2003.

[29] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. Xml schema part 1: Structures, second edition. *W3C*, October 2004.

[30] V. Tosic, K. Patel, and B. Pagurek. Wsol - web service offerings language. *Lecture Notes in Computer Science*, 2512/2002:57–67, 2002. ISSN 0302-9743.

[31] M. Wooldridge. Agent-based software engineering. *IEEE Proceedings Software Engineering*, 144(1):26–37, 1997.

[32] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet Auctionbot: A configurable auction server for human and software agents. *Second International Conference on Autonomous Agents*, May 1998.