

Service Level Agreement Management with Adaptive Coordination

Dominic Greenwood and Giosuè Vitaglione and Lukas Keller and Monique Calisti

Whitestein Technologies AG
Pestalozzistrasse, 24
CH-8032 Zürich, Switzerland
{dgr, gvi, lke, mca}@whitestein.com

Abstract

Service Level Agreement Management in the telecommunications domain consists of a set of mechanisms for provisioning and monitoring services according to requirements given by either a customer or provider. This paper presents an approach to adaptive coordination between customers requesting service via an SLA, and the provider who owns and provisions network resources. We deploy a set of software agents capable of automating negotiation between these parties, while applying provider policy and controlling admission of service requests onto the underlying network infrastructure. An architecture supporting this approach is described, as is a prototype of the full system deployed on a network simulator.

1. Introduction

In contemporary communication networks, many aspects of the interactions between service providers and their customers are regulated by static and relatively inflexible Service Level Agreements (SLA). Bound into the parameters of these contracts are details such as the number of links, network nodes (or access points) and the available capacity across end-to-end paths used to deliver content, perhaps at using a certain pricing model. What is all too often neglected is an account of the current network state and resource deployment balanced against stringent realtime constraints (resources being anything from physical network equipment to available capacity across a link). As a consequence networking performance and management tasks are strongly constrained, since the options available to dynamically accommodate offerings and balance the load across the network resources is very limited or even impossible.

This naturally results in inefficient utilization of network resources with suitably consequential losses in revenue.

The strong limitations of traditional approaches for service provisioning and SLA Management¹ (SLAM) are directly related to the lack of flexible and efficient methods/tools capable of proactively and dynamically supporting providers decisions. This is compounded by the difficulties encountered by human operators when attempting to consider all factors that influence the SLAM process and make decisions in real-time. This complexity includes the increasing number of actors in deregulated Telecom markets and the use of often heterogeneous technologies.

Considering these aspects, today's and tomorrow's evolving network scenarios require a management solution that uses both static and/or mobile software entities to collect and analytically process network state information to support the decisions taken by human operators and, when suitable or necessary, directly effect changes in network components.

We report on our solution to this requirement; a SLAM System that directly employs software agent technology to effect a proactive and flexible means of managing resources and SLAs in IP QoS based networks. We describe how agents are used to coordinate the decision process of determining the value of SLA parameters between customers and operators. We also show the use of a dedicated policy server agent to control this decision process.

This SLAM System has been developed to software prototype stage connected to a network simulator to evaluate behavioral response under varying network conditions, policies and customer requirements. This prototype, developed using the Living Systems® Technology Suite² [3], is briefly described.

The following section of the paper gives an overview of our position on Service Level Agreement Management and

Accepted for the International Conference on Networking and Services (ICNS'06), July 19-21, 2006, Silicon Valley, USA.

¹See the TeleManagement Forum technical program on SLAM at <http://www.nmf.org/browse.asp?catID=1690>

²For further information on LS/TS see: http://www.whitestein.com/pages/solutions/ls_ts.html.

the specific benefits brought about through the use of software agent technology. In this section we also define our notion of Service Access Control which governs the acceptance of a service provisioning request on the network infrastructure. Section 3 then describes our SLAM architecture, including the agent system, coordination model and policy model. Finally Section 4 provides a brief overview of our prototype implementation before closing the paper with conclusions.

2. Service Level Agreement Management

The basis for SLAM is the Service Level Agreement (SLA) [2] which is a documented agreement between a service provider and a service recipient and defines the basis of understanding between the two parties for delivery of a service. We define an SLA as containing clauses defining service type, service level, QoS parameters such as delay, loss and jitter, reservation information, pricing, etc. This SLA is reified into an SLS [1], which is technical mapping of the SLA that can be used by the Service Admission mechanisms of network infrastructure.

The objectives of our work, are to provide an agent-based framework for efficient and flexible Service Level Agreement Management in telecommunication networks. This is contingent on the capacity to negotiate and re-negotiate SLAs, to warn the customer of possible violations of the service guarantees and also to be able to reliably estimate the effect of SLA admissions or tear downs on the network properties in a short and midterm perspective in order to ensure a sound basis for negotiation. The task of estimating in advance whether the QoS guarantees of a service instance can be kept during the entire lifetime of that service is managed by Service Access Control (SAC) to clearly separate it from Call Admission Control (CAC) which denotes the behaviour of routers to reject certain calls in case where they are overloaded.

Software agent technology [6] [3] [4] is a means of creating autonomous, intelligent and social software assistants capable of supporting human decision-making. An agent comprehends and interacts with its environment and components present within that environment, e.g., humans, agents, services, etc. There is some evidence in the literature of software agents being applied to the SLAM domain, including [5] which deals with management and monitoring of SLAs.

Software agents in SLAM provide the means to support and automate process and decisional aspects of end users and/or service providers behaviour. In this respect they help to:

- Reduce the complexity of tasks including time constrained negotiation of SLA parameters, pricing schedules and even dynamic network re-configuration.

Achieved by implementing knowledge processing and analytical techniques that support and optimize the human decision making process. Complexity of control is pushed into software agents: controlling or modifying agents is easier and cheaper than changing the network and management stack.

- Provide a scalable network management approach with flexible task assignment across a population of agents
- Offer value added services, such as the integration of activities performed during SLA set up with other network management services, or the personalization and custom-tailoring of some tasks for which there is now no means of differentiating the way end users and services are monitored and maintained.

3. SLAM Architecture

The general architecture of the SLAM system is shown in Fig. 1. As mentioned, the key features of our approach to SLAM is the use software agents to efficiently and flexibly negotiate the specific details of SLAs, apply operator policy overlays to SLA decisioning, provision and monitor desired and specified Quality of Service (QoS) levels, automatic reporting and optimal deployment of network resources according to customer history models.

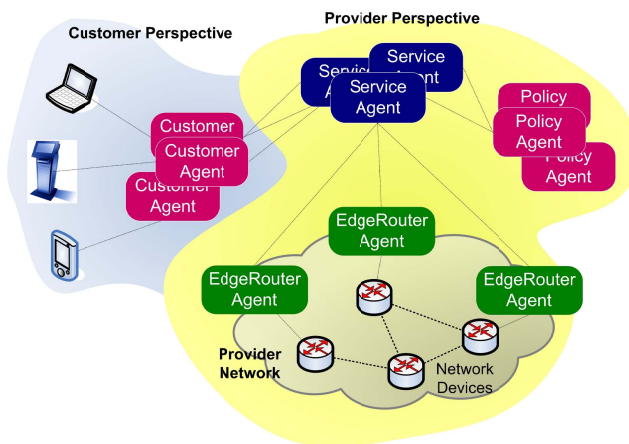


Figure 1. SLAM Architecture

From Fig. 1 we can see that SLAM consists of two perspectives. That of the customer and that of the provider. Each customer is represented by a *customer agent*, often located on a user device or item of network terminating equipment. These agents coordinate with the network provider's *service agents* to agree on the parameters of SLAs defining some service provisioning. The provider uses *policy agents*

to govern the decisioning aspect of the customer coordination and *edge router agents* to control Service Level Access (SAC) onto the core network. SAC is the mechanism by which a service is accepted or not for provisioning on a network infrastructure and is the key mechanism for maintaining preferred levels of QoS. Whenever a new request for service made (via a Service Level Specification), SAC is responsible for verifying whether the conditions of the request set can be satisfied with currently available network resources.

3.1. Agent System Model

The SLAM System is composed of several interacting software agents, as indicated in Fig. 2. Customer agents are unique in that they belong to the customer perspective, i.e., are essentially operated by customers from their local devices. The remainder of the agent classes are owned and operated by the service provider and reside within the enterprise boundary. Each class of agent can exist in multiplicity, as required according to topological, load or other distribution constraints.

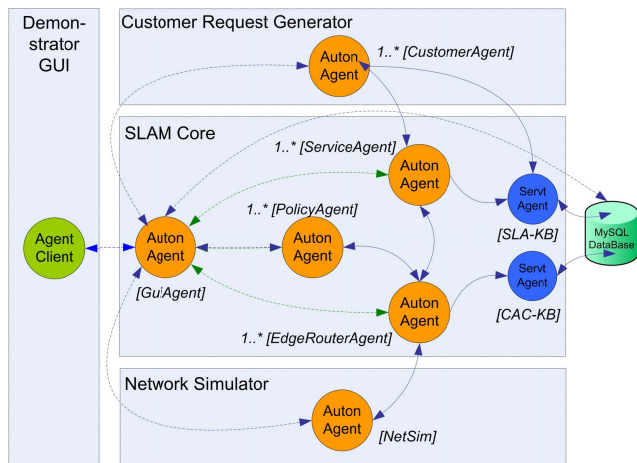


Figure 2. SLAM Agent System Model

The agent classes are:

- *Customer Agents*. These agents reside on customer devices and act to support their human owner in negotiating SLA setup and dynamic change according to specified requirements and constraints. In simulation mode, the customer agents can also mimic the expected behaviour of real customers by generating SLA requests according to a variety of selected algorithms. Thus simulated contracts can be established with static or dynamic transient characteristics that describe service requirements and financials for individual or clusters of customers. In this way, the effects of different load

conditions of groups of users with different user profiles can be simulated and evaluated.

- *Service Agents*. These agents are the operational heart of the SLAM System and logically reside between the service consumers (customers) and the service provider. Their primary responsibility is to negotiate the specific parameters contained within an SLA. Typical parameters include service type, service level (equates to quality class), pricing and if necessary fine tuned variables for bandwidth, delay, loss, jitter, etc. Service Agents must also verify all negotiated SLAs with the policy agents to ensure that no contracts or operational policies are violated.
- *Policy Agents*. The Policy Agents are responsible for administering provider policy. Each policy agent contains a filter chain which contains one or more sequentially connected filters. An incoming proposal (i.e., an agreement formed between customer and provider for the delivery of a service), is passed through each filter in turn, each of which has veto power to reject the proposal if certain conditions are not met. Filters can be added or removed at runtime and loopbacks triggered if necessary. Some examples of filters include verification that established customer contracts are not violated by a negotiated SLA, and filters that ensure that business rules defined by the provider are observed and applied appropriately to service provisioning.
- *Edge Router Agents*. The Edge Router Agents reside at the network edge and are responsible for interfacing between the SLAM service agents and the underlying network infrastructure. Their primary task is to manage Service Admission Control. This is the process by which a service provisioning specification (i.e., the SLS, transcribed from the original SLA), is checked against available network resources and either admitted or rejected accordingly. The edge router agents use operational constraints that define the levels of acceptable performance that must be maintained on the network infrastructure, e.g., acceptance thresholds to ensure a desired level of QoS is maintained across the network provisioning.
- *GUI Agent*. This agent simply acts as an interface between the GUI front-end and the SLAM Core agents.
- *NetSim Agent*. This agent is responsible for controlling the underlying network simulation over which the SLAM System can be tested. The simulator is a detailed, purpose built, system that uses a standard M/M/1/k queuing model for representing both routers and links. The simulator contains functional models

for Diffserv, RSVP and MPLS. We consider any further description of this model to be out of scope for this paper.

3.2. Primary SLAM Process

The high-level operational sequence of the SLAM System is as follows:

1. A customer agent formulates SLA detailing service provisioning request and sends to a provider service agent.
2. A service agent accepts the SLA request, converts into an SLS (Service Level Specification) and forwards the request to an edge router agent having applied policy from the policy agent as appropriate.
3. The receiving edge router agent and service agent negotiate the specific terms of the service provisioning based on SLA requirements and network resource availability.
4. Proposals are returned to the customer agent, who can accept or reject offers. Rejections typically trigger re-negotiation.

3.3. Adaptive Coordination Model

The SLAM System employs several mechanisms to increase the efficiency of the service provisioning by fully exploiting the available capacities of the network. Most of the mechanisms are based on a better coordination of the potentially conflicting interests of customers and providers.

Although we can assume that in a longer term perspective, every provider essentially aims at increasing its revenue, its immediate goals can be manifold: optimizing network utilization, reducing operational costs, and maximizing customer satisfaction (minimizes the SLA preemption, violation and rejection rates) in order to retain customers and reduce churn rate.

Customers simply want the best possible service for the cheapest possible price. However, there are many different classes of customers, which are best satisfied with a different balance between prices and quality of the service.

SLAM performs service access negotiation taking into account the status of the network, and exploiting availability of resources that would remain unused with a static network behaviour.

A number of interaction types have been identified that increase the efficiency and flexibility of the system:

- *Sequential Coordination.* If the service properties can not be guaranteed as requested by the customer, the

system (in particular the Service Agent) sends alternative proposal(s) that can be satisfied by the current status of the network and whose content is as close to the conditions of the original request as possible. If such proposal is rejected by the customer agent, a limited number of additional proposals can be formulated by the system and sent to the customer.

- *Concurrent Coordination.* Service requests coming from multiple users are no longer treated individually but several of them arrived in the same time interval are evaluated together. In case of a lack of resources, those that seem to be the most beneficial are treated preferentially. Concurrent coordination allows better improved optimisation of network resources.
- *Deferred Service Setup.* If the customer agent considers as not satisfying the received offers, he can decide to register a standing request at the Service Agent together with an expiration date and an upper price limit. If within its lifetime the system can provide the required quality to the specified costs, the user will be notified. Another related mechanism is the advanced reservation of resources, in the sense that services are not set up immediately after the conclusion of the SLA agreement, but at some selectable moment in the future. This would require a strict scheme for resource reservation or a predictive network state estimations mechanism.
- *On-line Re-negotiation.* In some cases, re-negotiation of a service being provided might free resources to satisfy multiple service requests, possibly improving the overall efficiency. In this kind of interaction, the Service Agent initiates a negotiation with a Customer Agent who is being provisioned a service. This interaction can result into a new agreement with modified service parameters or even service type.
- *Push Service Offering.* The Service Agent sends Special Offers to the Customer Agent according to the status of the network. The system could automatically, or assist to, estimate the impact on providers revenue and network usage of special offers like temporary price reductions or combination-offers.

To date we have concentrated our efforts on the implementation of a viable sequential coordination mechanism. Some of the more advanced concepts that have not been implemented will be discussed at the end of this chapter. A schema describing the agent coordination interactions before the setup of a service is shown in Fig. 3:

The Service Agent maintains a map containing default traffic parameters (DSCP, preemption priority, RSVP pool)

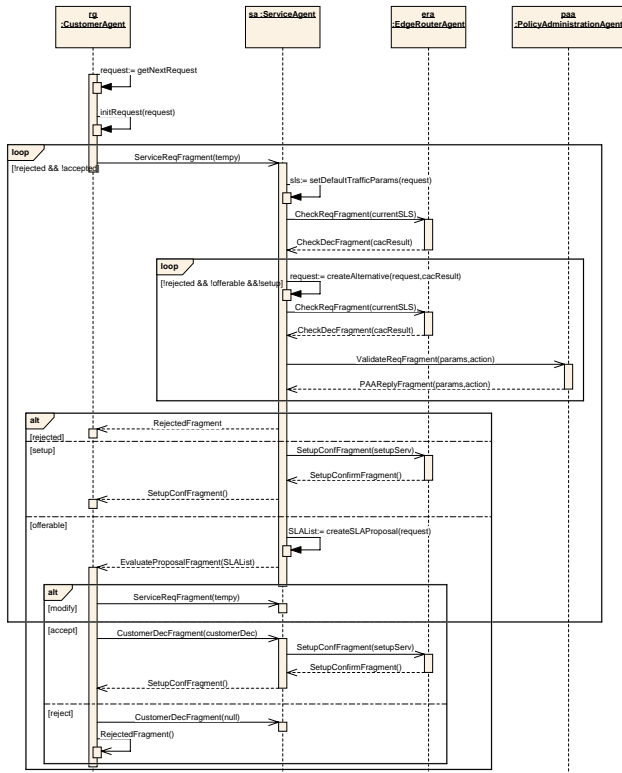


Figure 3. SLAM Coordination Interactions

for the different service types. Upon reception of a new request, the Service Agent assigns it the default parameters and submits it to the Edge Router Agent for SAC. For all those LSPs to which the request could be assigned, the SAC procedure on the Edge Router Agent now returns precise information on the quality of the flows of the tested traffic parameter configuration (CACData cacResult) and for the others it indicates the reason why the request can not be routed over them. The Service Agent uses then this information to decide on the next actions it will take. When receiving the SAC results, the Service Agent can opt for different actions:

- *Setup*. If, with the default parameter settings, there is a path where the quality is good enough for the required service, it can be setup directly and the customer is notified of its availability.
- *Offer*. If it is not sure whether the quality of the proposals is good enough or if wished by the user, the Service Agent selects a subset of the Edge Routers proposals and offers them to the customer. Depending on the providers policy, the offers may be editable (offerMod), in which case the customer can modify the service requirements of an SLA and resubmit it to the SA, or non-modifiable. In the latter case (offer), the customer can only either accept one of the proposals

or reject the whole set and stop the interaction.

- *Continue*. Considering the results received for previously tested parameter combinations, it can test the quality and resource availability for different than the default traffic parameters until it has either found one (or several) combination(s) that match(es) the services requirements and can be proposed to the customer or until all meaningful combinations have been tried.
- *Reject*. If the Service Agent is convinced, that the request can currently not be served, it is rejected.

Which of these actions will be taken under which pre-conditions depends essentially on the configuration of the Service Agent. When proposals are sent back to the customer, the customer can either (1) Select the best and ask the Service Agent to set it up (accept). If there is no satisfying offer, he can (2) Reject them and stop, such that the coordination ends without service setup (stop) or, if allowed by the provider, (3) Specify other quality requirements and resubmit the request to the Service Agent to obtain more proposals (continue).

3.4. Policy Enforcement

Whereas the conformance of the particular service parameters to the customers requirements is checked by the service agent, it is the responsibility of the policy agent to enforce the providers goals. Since the service agent double checks all important decisions first with the policy agent, the latter is able to control the behaviour of the entire SLAM system.

The definition and enforcement of declarative generic administration policies is still a research topic and out of scope for this paper. Therefore we focus on the design and implementation of a set of simple procedural pluggable policies. Policy enforcement is performed through the interaction between the service agent and the policy agent. Following interactions with a customer agent and an edge-router agent, a service agent evaluates the SAC results, and proposes the next action to be taken, i.e. further proposals to the customer agent. These are passed to the policy agent to obtain approval to proceed with the negotiation. The policy agent can send back a subset of the proposals to the service agent, imposing constraints according to the goals of the provider expressed in the policies.

3.4.1 Policy Filter Chain

The policy agent is composed by a chain of policy filters which allow variable composition of policies.

Provider's business goals are mapped to one or more policy filters, that take the proposal as input, check its confor-

mance to the filters rules and if necessary adapts the proposals content accordingly. Depending on the nature of the goal in case of offer and setup, the filter can attribute a scoring value to each proposal that indicates how well the configuration respects the filters goal. Above a certain threshold it can veto the proposal, which is discarded.

Filters can be chained, such that the output of a first filter serves as input of a second one, the final output will be submitted to and executed by a service agent. If a proposal is vetoed by a filter, it is not even passed for evaluation to the following filters in the chain (sine qua non conditions).

The filters can also change the proposal sent by the SA. If, for example, a service agent proposes to offer a certain set of service configurations to the customer, the filters on the policy agent can disregard the advice and decide to change the quality provided according to a certain policy. Note that if some filter changes the action, the new action proposal must re-traverse the filter chain from the beginning to assure its conformance with all the providers goals.

4. Implementation of the SLAM System

The SLAM System is implemented using the Living Systems[®]Technology Suite (LS/TS). This is a commercially available, industry-strength software agent platform, compliant with both J2SE and J2EE and with extensive Eclipse based development and deployment tool support.

All SLAM agents are implemented in Java as LS/TS agents with Petri net based behavioral logic. Communication is directly supported by the agent platform and the agents may be distributed across multiple network devices. The LS/TS administration tool allows the entire agent population, databases, interactions and events to be monitored in real time with intervention policies deployable as required.

A snapshot from the GUI front-end of the prototype SLAM System is shown in Fig. 4.

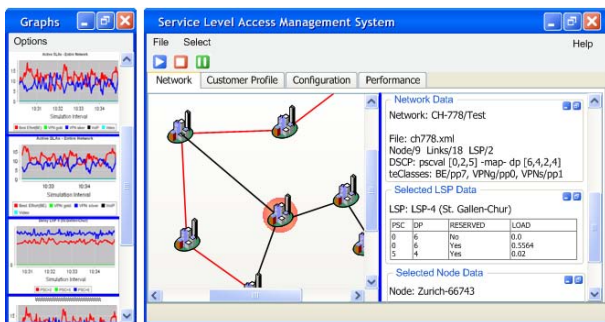


Figure 4. SLAM Implementation GUI

This snapshot shows a network topology that can be modified in real-time, with data on the right concerning the

particular network in live simulation and selected LSP (Label Switched Path). On the left are several charts reporting on performance of the simulation in terms of selected parameters such as active SLAs, the jitter, loss and delay associated with any given LSP or Node in the network simulator, and behavior of the individual agents during decision and coordination interaction.

5. Conclusions

This paper has presented an approach to achieving adaptive coordination between customers of telecommunications service providers and their providers, expressed using SLAs. The reported system consists of a set of interacting software agents that automate negotiation between the parties, while allowing the application of policy controls, to control service admission onto an underlying network infrastructure. The current prototype implementation can be deployed and integrated within network management platforms. A future version of the system will consist of agents that use of the new Goal Based Planning feature of Living Systems[®]Technology Suite to enhance their behavioral characteristics.

References

- [1] D. Goderis et al. Service level specification semantics, parameters and negotiation requirements. IEEE Internet Draft, 2001.
- [2] John Evans and Clarence Filstils. Deploying diffserv at the network edge for tight slas, part 2. *IEEE Internet Computing*, 8(2):61–69, 2004.
- [3] Giovanni Rimassa, Monique Calisti, and Martin Kernland. Living systems[®]technology suite. *Whitestein Series in Software Agent Technologies Systems: Software Agent-Based Applications, Platforms and Development Kits*, 2005.
- [4] I. Trencansky and R. Cervenka. Agent Modelling Language (AML): A comprehensive approach to modelling mas. *Informatica*, 29(4):391–400, 2005.
- [5] F. De Turck, S. Vanhastel, P. Backx, B. B. Duyburgh, and P. Demeester. Design of a generic architecture for service management and monitoring of service level agreements through distributed intelligent agents. In *Proceedings of IEEE Intelligent Network Workshop*, pages 50–57, 2001.
- [6] Michael J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2001.