

Self-Adaptive Configuration of an Overlay-Based Application

Alexandre Sztajnberg², Orlando Loques¹

¹Instituto de Computação, Universidade Federal Fluminense (UFF)

²DICC/IME e PEL/FEN, Universidade do Estado do Rio de Janeiro (UERJ)

loques@ic.uff.br, alexszt@ime.uerj.br

Abstract. *This paper discusses the use of architectural contracts to specify the configuration of components and the resource requirements for a videoconference overlay-based application. In addition, these contracts can guide configuration adaptations regarding the overlay network composed by videoconference reflectors and user terminals. The approach allows to improve the quality of the terminal connections, to provide self-repair characteristics to the overlay network, and to dynamically manage the set of reflectors in order to maintain the required application quality.*

1. Introduction

Our investigation is centered on the CR-RIO framework [Loq04] that includes concepts and mechanisms that facilitate the specification, deployment and management of adaptive applications. We developed a series of experiments, with a self-adaptive version of a videoconference application, focusing on how the CR-RIO can facilitate the design and deployment of applications running on constantly changing environments. The experiments were performed in the context of the Giga-RNP project (www.projetogiga.org.br).

2. CR-RIO Basics

The CR-RIO framework is centered on an architectural model and uses an ADL, CBabel, and a contract description language to express applications non-functional requirements [Loq04]. Based on these elements a supporting infrastructure was conceived to: (i) parse the contract specifications and store them as meta-level data, (ii) provide reflective and adaptive mechanisms, which allows adapting the application's configuration, and (iii) provide a set of mechanisms to interpret, impose and manage these contracts.

Contracts. A contract describes at design time non-functional aspects of the application, specifying how supporting resources should be used in running time, and the acceptable variations on the availability of these resources. A typical contract has the following elements: (a) Categories, which describe, in an abstract level, properties of resources, or specific non-functional aspects; (b) Profiles, which quantify the properties of a given Category; (c) a set of Services, where each Service defines an acceptable architecture configuration constrained by a list of profiles; (d) a Negotiation clause that describes a policy, defined by a transition system, which establishes a particular strategy to select and deploy one of the Services.

Support Infrastructure. Architectures described in our ADL are mapped to an object model [Cor05]. This model is reflected on a meta-level repository, which

maintains the configuration information of the architecture that can be queried and updated during the application's lifetime. This repository also includes the representation of the contracts and maintains information regarding resources (devices, services, applications, etc.) that are of interest of the application. Based on the contracts, which use the information contained in the repository, a supporting infrastructure is used to manage the architectural configurations. This infrastructure is composed by a standard set of components (please refer to [Cor05] for details).

Dynamic Selection of Resources. Two additional services complement CR-RIO's infrastructure: Discovery and Context [Car06]. The Context service encapsulates the monitoring of resources involved in a contract, and is provided by a XML based interface. The Discovery service can identify and select appropriate available component instances, based on the informed component class and resource constraints specified in the profiles. The selection of the most appropriate component can be accomplished by describing an utility function (incorporated into the contract text), which is also parametrized by the profiles and by weights associated to the relevant properties. In that way, a contract can specify components to be dynamically selected and integrated to the current service configuration or to the service being deployed

3. Videoconference Application

The basic scenario for the videoconference application comprises (i) an arbitrary number of users distributed over a network such as the Internet, with (ii) an overlay network used to provide multipoint communication through point-to-point connections. The overlay network architecture is formed by reflectors, interconnected by communication connectors, which relay the audio and video flows, as described in [Gar04].

In our approach different portions of the application architecture can be handled by specific contracts that work in synergy, aiming to meet the specified overall requirements. Regarding the videoconference application some contracts were used:

Overlay setup: This contract describes the basic reflector topology and the required resource (CPU, memory, link bandwidth and delay) levels to deploy the service. This basic contract is used to setup the initial configuration of the reflector's overlay network.

Overlay network selection. In our experiment we worked with two different overlay networks: one of them used links from the commodity Internet and the other Giga bit dedicated channels; both were composed by three reflectors. We configured each set of reflectors

with a specific set of profiles, adequate to indicate their non-functional requirements. The idea is that “overlay providers” would offer differentiated access services, and the user, using a contract and consistent profiles, can specify a policy for choosing one of those services. Say, the *viaInternet* service should be preferably used – given that the non-functional requirements are being met. In the negative case, the *viaGIGA* service, with better resources, but more expensive, is negotiated and is made the current service. This policy would be defined in the negotiation clause in the contract.

Terminal connection. Each terminal is treated as an independent portion of the architecture and is associated to a terminal contract, specified to cover the involved requirements: (i) media quality; (ii) communication characteristics; and (iii) videoconference session management. The instantiation of a terminal, the terminal-reflector connection, and the dynamic selection of the most appropriate reflector can also be ruled by the contract. The selection of the best reflector uses a utility function that describes the user preferences for accessing the application.

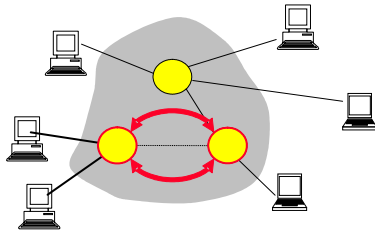


Figure 1. Link self-repair case

Self-repair. In the case of saturation or failure of a link connecting two reflectors, the interconnection can be reconfigured using an alternative link (Figure 1). (Due to the space constraint we detail only this contract).

```

contract {
  service { link refUFF to refUERJ by
    con = select*(comCon, futill1) with prfP;
  } primeLink;

  service { link refUFF to refUERJ by
    con = select*(comCon, futill2) with prfA;
  } altLink;

  negotiation {
    not primeLink -> altLink;
    altLink -> primeLink;
    not altLink -> out-of-service;
  }
} cAltOvl;

```

Figura 2. Link self-repair contract

A contract, such as in Figure 2, can describe the possible alternative configuration of the link between two reflectors. By using different communication connectors, independent routes are used in each service. In the example two services are described: *primeLink* and *altLink*. If the preferred service, (*primeLink*, that is, the preferred, high-quality, link) is not available or cannot be maintained, the architecture is reconfigured to use the alternative service, *altLink*, (the alternative, with lower quality, link). If the preferred service becomes available it is established again. If neither service can be deployed this part of the architecture becomes unavailable, fact also reflected in the whole overlay network. This policy is described in the negotiation clause of the contract.

In the *primeLink*, for instance, it is described that the *refUFF* and *refUERJ* reflectors will be linked by the *con* connector. The *con=select*(comCon, futill1)* construction states that when deploying the contract, a connector from the *comCon* class must be dynamically selected and assigned to the *con* reference. Furthermore, the selected connector must comply with the *prfP* profile (with high-quality values in this case), and among the suitable connector instances the best one will be selected according to the *futill* utility function. The *con* reference to the selected connector can then be used in the configuration operations.

This selection procedure is supported by the Discovery and Context infrastructure services. Also, the *select** primitive periodically checks if the currently selected connector is still the best choice. If this is not the case, *con* is atomically updated with the better one. If a connector with the required constraints is really not available, the *primLink* service is invalidated, leading to a new negotiation round trying to deploy the *altLink* service.

4. Final Remarks

Our intention is to improve the framework in order to cater for more dynamic configuration management techniques. The framework (contract syntax and support levels) already provides the capability of adding or removing components. In the videoconference application, this capability can be used, in an *ad hoc* fashion, to expand or reduce the set of reflectors composing the overlay network, keeping the required quality level, while making a rational use of the available resources. In a next step, we intend to provide a built-in quality inspection mechanism, able to check individual properties of the components of a given set, intending to identify specific components that do not comply (eg, exceed or do not supply) with the application requirements. This mechanism can be used to pinpoint components to be dynamically configured, providing a useful abstraction in the engineering of adaptive applications, such those using dynamic overlay networks.

In the context of a GIGA-RNP project, we are investigating (with a team from UFPA) the use of the framework to manage customized overlay networks, combining MPLS tags, QoS configurations and optical lambdas.

References

- [Cor05] Corradi, A. S., “A framework to support non-functional requirements for high-level services”, MSc. Dissertation, IC, UFF, 2005. (in Portuguese)
- [Car06] Cardoso, L. X. T., Sztajnberg, A.; Loques, O., “Self-adaptive Applications using ADL Contracts”, LNCS, Heidelberg, v. 3996, p. 87-101, 2006.
- [Gar04] Garlan, D., Cheng, S., et al., “Rainbow: architecture-based self adaptation with reusable infrastructure”, IEEE Computer, Vol. 37, No. 10, October, 2004.
- [Loq04] Loques, O., Sztajnberg, A., et al., “A contract-based approach to describe and deploy non-functional adaptations in software architectures”. JBCS, Vol. 10, No. 1, pp. 5-18, July, 2004.