Chapter 8
Logical Methods for Self–Configuration of Network Devices

Sylvain Hallé
Université du Québec à Chicoutimi, Canada

Roger Villemaire
Université du Québec à Montréal, Canada

Omar Cherkaoui
Université du Québec à Montréal, Canada

ABSTRACT
The goal of self-configuration consists of providing appropriate values for parameters that modulate the behaviour of a device. In this chapter, self-configuration is studied from a mathematical logic point of view. In contrast with imperative means of generating configurations, characterized by scripts and templates, the use of declarative languages such as propositional or first-order logic is argued. In that setting, device configurations become models of particular logical formulæ, which can be generated using constraint solvers without any rigid scripting or user intervention.

INTRODUCTION

Despite the tremendous development of network services and functionalities over the years, the configuration and deployment of network elements such as routers and switches remains a mostly manual task. An intricate knowledge of each devices’ and services’ inner workings and dependencies between configuration parameters is required from the network engineer in order to successfully run even basic use cases. The addition of a new device or the deployment of a new service to an existing infrastructure requires repetitive but careful manipulation of multiple configuration parameters on many elements, and even such a cautious work can spawn unpredicted side effects that are discovered by trial and error.

The wish for systems behaving automatically can be traced back to the foundations of control theory (Maxwell, 1867). The application of such
principles to software systems started being studied in the late 20th century (Williams & Nayak, 1996). The advent of increasingly powerful devices showing complex behaviours and myriad configuration states led researchers to raise a “Call to action” for the development of systems following radically different design principles. The study of self-configuration and self-healing as a field in itself got its kick-start with IBM’s Autonomic Computing initiative in the early 2000s. Rather than relying on manual configuration, inter-operation, and repair of computer systems, Autonomic Computing pushes forward the concept of mimicking properties of the autonomic nervous system. Among the so-called “self-*” capabilities that should be exhibited by such systems, we mention self-configuration, self-healing, self-deployment, self-management, and so on.

The application of the Autonomic Computing and Networking (ACN) paradigm (Parashar & Hariri, 2004) to computer networks offers a promising means to release the burden of knowledge and tedious manipulations currently needed from engineers. It has long been argued that Internet Service Provider environments are becoming too complex to manage on an individual system-to-system basis (Melcher & Mitchell, 2004). By definition, self-governed devices can automatically modulate their behaviour in reaction to inconsistencies in the state of the network, or to changes in its topology or management policies.

ACN Through Self-Configuration

In this chapter, we focus on the implementation of self-configuration through formal methods. (Narain, 2004) describes a typical self-configuration architecture, called a service grammar, shown in Figure 1. Under such an architecture, specific system requirements, expressed in plain English, are first translated into some machine-readable notation, called a requirements language (1). This language serves as a reference, or goal, that must be reached by the autonomic system. This language is first fed to a synthesis engine (3), whose task is to generate a first instance of a system configuration satisfying the requirements (4). This configuration is then applied to the actual devices (5), possibly with the help of some vendor-specific adaptor that transforms the formal system configuration into actual commands and parameters.

Conversely, the devices’ configurations are periodically retrieved to update the formal system configuration. Changes to the configuration are detected by the diagnosis engine (6), which relays them back to the synthesis engine (7). Should some of the changes invalidate the requirements, the synthesis engine regenerates a new, valid system configuration and sends any modifications back to the appropriate devices.

It shall be noted that this architecture consists only of a few minor additions to the core cycle 4–6–7, which is nothing but a feedback loop in the traditional, control-theoretic sense (Hellerstein et al., 2004). Hence, as such, self-configuration can be seen as an application of control theory to information systems, and in particular network devices.

In turn, other autonomic, self-* properties are partly subsumed by the previous architecture. For example, self-deployment is achieved by iterating through the system a first time, starting from an empty configuration. Similarly, self-healing is a particular case of self-configuration, where harmful modifications to the devices’ configurations are detected by the diagnosis engine, and corrected through the generation of a new configuration that re-establishes the network’s conformance to its requirements.

ACN Through Self-Configuration

This framework supposes that system requirements be expressed in a way that makes possible reasoning and computation about them. Although the service grammar architecture provides an