A Constraint-Based Approach to Diagnosing Software Problems in Computer Networks

Daniel Sabin, Mihaela Sabin, Robert D. Russell and Eugene C. Freuder

Department of Computer Science, University of New Hampshire, Durham, NH 03824 USA

Abstract. Distributed software problems can be particularly mystifying to diagnose, for both system users and system administrators. Model-based diagnosis methods that have been more commonly applied to physical systems can be brought to bear on such software systems. A prototype system has been developed for diagnosing problems in software that controls computer networks. Our approach divides this software into its natural hierarchy of layers, subdividing each layer into three separately modeled components: the interface to the layer above on the same machine, the protocol to the same layer on a remote machine, and the configuration. For each component knowledge is naturally represented in the form of constraints. User interaction modeling is accomplished through the introduction of constraints representing user assumptions, the finite-state machine specification of a protocol is translated to a standard CSP representation and configuration tasks are modeled as dynamic CSPs. Diagnosis is viewed as a partial constraint satisfaction problem (PCSP). A PCSP algorithm has been adapted for use as a diagnostic engine. This paper presents a case study illustrating the diagnosis of some problems involving the widely used FTP and DNS network software.

1 Introduction

One of the fundamental problems confronting users and managers of computer networks today is the diagnosis of problems arising within the network itself. The symptoms produced by such problems are often baffling since they are so unpredictable and so unrelated to the task for which the network is being used. Furthermore, the error messages from the system are usually so general and vague that little can be gleaned from them as to the exact cause of (and hence the fix to) the problem. Diagnosing under these circumstances is currently more art than science [14]. The situation has been summarized in a cartoon that pictures a visitor to a computing site staring at a swami sitting cross-legged in the corner, and receiving the explanation: "That's our network guru".

Techniques for model-based diagnosis have been used successfully in the diagnosis of physical systems [10]. We have applied and extended this approach to computer network software. We consider this software to be constructed in a *hierarchy of layers* fashion as described in the ISO OSI Reference Model [12].

We subdivide each layer into three separately modeled components: the interface to the layer above on the same machine (or to the user, in the case of the application layer), the protocol to the same layer on a remote machine, and the configuration. This decomposition is developed further in the paper.

Our approach considers diagnosis as a dynamic partial constraint satisfaction problem. Activity constraints are used to interface the model with the "real world" of the network, allowing the model to dynamically obtain data from the network and to use that data to change the problem as the search for a solution progresses. The partial solutions discovered by our system constitute the diagnosis.

The next section concentrates on individual components, presenting examples of problems from widely used *File Transfer Protocol* (FTP) and *Domain Name Service* (DNS) software. Sect. 3 presents the theoretical background for our approach. Sect. 4 gives some details on how problems are represented in our system, presents the actual dynamic partial constraint satisfaction algorithm and shows how one of the sample problems, previously presented in Sect. 2, is actually diagnosed by this algorithm.

This work concentrates on modeling the network *infrastructure* itself. We believe it gives us a solid foundation for understanding basic problems that occur at all levels in information networks, and that the techniques we have developed are applicable at higher levels as well.

2 FTP Case Study

The application protocol chosen for exemplification is FTP, described in RFC 959 [19]. It provides interactive file transfer and relies on Transmission Control Protocol (TCP) – a transport protocol in the TCP/IP hierarchy of protocols [4].

We decompose the problem domain into three components, each of which will be modeled separately, as shown in Fig. 1.

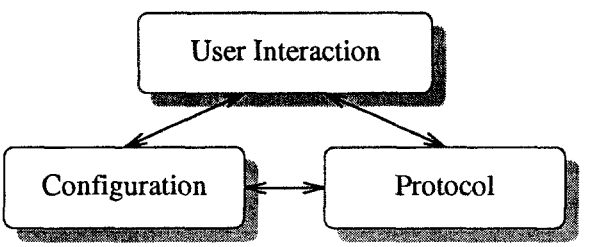

Fig. 1. Problem domain decomposition

1. The *User Interaction* component deals with commands given by the human user and the expectations that the user has about the system response to these commands.
2. The *Protocol* component is really the *network* software, and will later be further decomposed into layers corresponding to the actual implementation of the network software. For now it represents simply all *active software*.