1-1-1997

Generating diagnostic tools for network fault management

Mihaela C. Sabin
University of New Hampshire, Manchester, mihaela.sabin@unh.edu

Robert D. Russell
University of New Hampshire, Manchester, Robert.Russell@unh.edu

Eugene C. Freuder
University of New Hampshire, Manchester

Recommended Citation
Mihaela Sabin, Robert D Russell, and Eugene C Freuder, Generating diagnostic tools for network fault management, Integrated Network Management V, Springer US, 1997, pp. 700–711.

Generating Diagnostic Tools for Network Fault Management

Mihaela Sabin, Robert D. Russell, and Eugene C. Freuder
Department of Computer Science
University of New Hampshire
Durham, NH 03824
mcs,rdr,ecf@cs.unh.edu

Abstract

Today's network management applications mainly collect and display information, while providing limited information processing and problem-solving capabilities. A number of different knowledge-based approaches have been proposed to correct this deficiency, evolving from rule-based systems through case-based systems, to more recent model-based systems. Part of this evolution has been the recognition of the importance of *constraints* in a management context. This makes possible the assimilation into network management of a mature, theoretically developed technology from artificial intelligence, namely, the *constraint satisfaction problem* (CSP). In this paper we investigate the role of constraints in manipulating management data, and give an example of the use of the constraint satisfaction framework in diagnosing problems arising with Internet domain name service configurations. We also present ADNET, a system for automatically constructing C++ diagnostic programs from a model written in a simple modeling language.

Keywords

Network fault management, configuration fault management, model-based diagnosis, constraint satisfaction, diagnostic tools.

1 INTRODUCTION

Today's network management applications mainly collect and display information, while providing limited information processing and problem-solving capabilities. The technological stages in managing network data classify network management applications into three distinct categories, suggestively characterized in (Rose 1993) as "browsers, mappers, and (very few) thinkers". Two essential factors explain the lack of the "thinkers": (1) the heterogeneity of managed objects which themselves are not problem solvers, and (2) the missing formalism and, consequently, technology for relating MIB modules within a configured, operational network. Thus, beyond the technical aspects of accessing formally defined managed objects through management protocols, an indispensable task is the semantic interpretation of this extremely heterogeneous data (Meyer *et al.* 1995): "Since the

semantic heterogeneity of managed data has grown explosively in recent years, the task of developing meaningful applications has grown more onerous". To address the problem of effective management applications, a framework for management applications should be centered around a global network model which explains the flow of data or the occurrence of events in the network with regard to some functions performed by the network.

In recent years, more knowledge-based systems developed in the network management domain have adopted the *model-based* approach for representing and reasoning about management information. In this approach, the building blocks of the network model are the elementary structural and behavioral descriptions which define the network components: devices, links, services. The network model, however, describes not only its constituents, but also how they relate to each other. The model-based reasoning paradigm has been successfully applied in the diagnosis task (Hamscher *et al.* 1992).

In this paper we propose a *constraint-based modeling and problem-solving* approach to network fault management. The approach is based on the concept of *constraints* which capture the structure and behavior of the network to be diagnosed, and which represent relations among network components. The network is found faulty if some constraints cannot be satisfied, in which case the violated constraints are precise indicators of the cause of the fault. The main contributions of the paper are:

- to show how an existing technology for reasoning with constraints can be applied to network management,
- to introduce a declarative modeling language,
- to present a system for automating the synthesis of special purpose diagnosticians, and
- to illustrate how these can be applied in a sample model of a high-level network service.

This paper is organized as follows. In the next section we define the constraint-based approach to the diagnosis problem, and the algorithmic solution to it. Section 3 describes the general structure of a diagnostician implementing this approach. Section 4 presents the architecture of ADNET, a system that builds these diagnosticians. Based on the example described in Section 5, Section 6 illustrates the ADNET modeling language. In Section 7 we briefly review existing knowledge-based approaches to network management, and in the last section we summarize our results.

2 DIAGNOSIS AS CONSTRAINT SATISFACTION

The constraint satisfaction problem (CSP) paradigm has proved its applicability in various areas of artificial intelligence, such as design, configuration, simulation, scheduling, and diagnosis. Its success has been assured by both the simplicity of formulating the problem, and the diversity of continually improved algorithms to solve it. In a constraint satisfaction model of a diagnosis problem, the *constraints* capture the structure and behavior of the system to be modeled, and represent relations among the attributes of the system components. A component attribute is modeled as a CSP *variable*, characterized by possible *values* to which the variable can be instantiated, according to either the design specifications or observation measurements on the modeled attribute. The diagnosed system is found faulty if some constraints cannot be satisfied, in which case the violated constraints are precise indicators of the cause of the fault.

We formulate the diagnosis problem as a partial CSP (PCSP) (Freuder and Wallace 1992), where the partial solutions stand for the minimal sets, under set inclusion, of violated constraints, also called minimal diagnoses (Sabin *et al.* 1995). For synthesis tasks such as configuration, the constraint problem is of a more dynamic nature. (Mittal and Falkenhainer 1990) defines the dynamic CSP (DCSP) formalism to take into account conditional activation of those parts of the CSP (variables and constraints) which are relevant to the current configuration decisions. However, we have adapted Mittal's approach in two aspects, in order to deal with the task of diagnosing configuration problems, rather than configuration itself (Sabin *et al.* 1995). First, the domains of values are not restricted to predefined sets of values, instead they can be acquired at search time by observing the network. Second, the DCSP which models the configuration of some network service is solved as a partial CSP, where the partial solutions leave minimal sets of constraints unsatisfied. These partial solutions capture the inconsistencies between actual observations and model predictions, and hence pinpoint the faults in the diagnosed system.

Combinations of branch-and-bound and CSP techniques have been used in algorithms that search for a solution that leaves *minimum-cardinality* sets of constraints unsatisfied (Freuder and Wallace 1992). We have adapted one of these algorithms to search for solutions with *minimal sets* of unsatisfied constraints, in order to provide a more comprehensive explanation of the possible faults. The algorithm keeps track of the best solutions found during the search, in the sense that any proper superset of these solutions is discarded, and any solution is replaced by its proper subset when such a subset is found. The algorithm serves as the inference engine of the diagnostic tool described next.

3 THE GENERATED DIAGNOSTIC TOOL

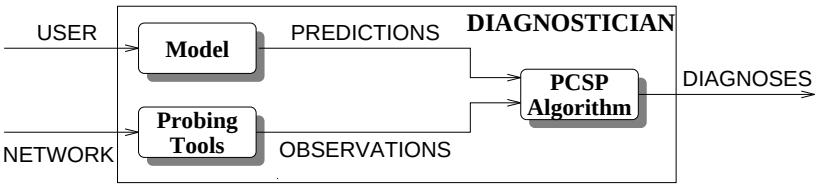

Figure 1 Architectural description of a diagnostician for network services

Automatic diagnosis employs a diagnostic engine which outputs the expected diagnoses based on the *predictions* of the model of the system to be diagnosed, and the *observations*, or measurements, performed during the system operation (Figure 1). Measuring the actual system behavior can be fully automated by incorporating probing or monitoring tools in this diagnosis scheme. The modeling descriptions are expressed in a form congenial to the human network manager, and embed only declarative knowledge, as found for example in system specifications manuals. The user of the diagnostician is shielded from the details of how the information is to be used, or what algorithms process this information. The model of correct (and possibly faulty) behavior of the network service is expressed in CSP terms, as explained in Section 6. The PCSP algorithm for computing the minimal diagnoses is used to check the consistency between the model predictions and system observations.

4 A SYSTEM FOR GENERATING DIAGNOSTIC TOOLS

The diagnosis scheme implemented in a diagnostician can be further automated if specialized diagnosticians are automatically generated to handle different categories of problems. For example, the model component in Figure 1 can describe network service problems in one of the following categories:

- user interaction problems, when a network service is improperly used,
- protocol operation problems caused by incompatibilities between the end-systems on which the protocol operates, and
- configuration problems, when the network service configuration contains missing or conflicting information.

Each model description is compiled into a C++ diagnostic program that handles the types of problems described in the model. The CSP formulation for each of these groups of problems has been detailed in (Sabin *et al.* 1995). In this paper we focus on the prototype system that automates the construction of constraint-based diagnosticians.

The automatic diagnosis system for network services (ADNET) constructs constraint-based diagnostician programs in C++ from a library of network probing tools, a library of constraint-based diagnosis tools, and a model. Figure 2 outlines the ADNET architecture. The model is written in the ADNET constraint-based modeling language. The resulting diagnostician is compiled and linked with a library of network probing tools and a library of constraint-based reasoning tools, to form an operational diagnostician. We describe each of the ADNET components next.

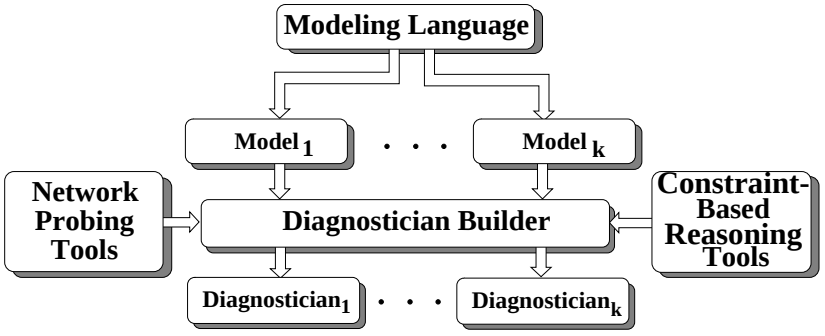

Figure 2 The ADNET architecture for the automatic construction of diagnosticians

The *models* are written in the ADNET modeling language and formulate a standard or dynamic CSP corresponding to those aspects of a network service which are to be diagnosed. The ADNET modeling language has the advantages of constraint programming in general, namely, the language is:

- declarative: stating the constraints does not require the user to envision how the information is to be used,
- natural: constraints are expressed in a form congenial to the user,
- efficient: heuristics and inference methods can mitigate the problems of combinatorial search.

The *constraint-based reasoning tools* form a C++ library that comprises different CSP algorithms and heuristics used to solve the constraint-based diagnosis problem. In the current implementation of ADNET, the diagnostician builder is programmed to use only the branch and bound algorithm for solving dynamic partial CSPs, discussed in Section 2. Maintaining a repository of CSP techniques has the advantage of enabling another interesting developing direction, namely, of tailoring the generated diagnosticians to the problem at hand to meet better efficiency requirements.

The *network probing tools* perform two functions in the ADNET architecture. First, they can provide the model with complete domains of values, which have been previously stored in the network at network configuration time. Tools that dynamically probe the network configuration parameters save the user from a preliminary phase of manually gathering this modeling data, and, more importantly, guarantee that the model always accurately reflects the current configuration values being used in the real network. Second, the probing tools can collect observational information for those CSP variables whose values depend on the actual network operation. Both features are available in the ADNET modeling language through the variable declaration construct. A variable declaration includes the function call that probes the network and gives the expected values (predefined as configuration data, or currently observed in the running network). Unlike modeling, probing the network is a device-dependent task. Some probing is general enough to be applied to a wide variety of networks, such as the Internet `ping` program to test the reachability of another site on the network. Other probing programs may be more specific to the type of the network resource to be examined. Thus, the ADNET system provides for both user-defined and system predefined probing functions.

5 AN EXAMPLE: DIAGNOSING DNS CONFIGURATIONS

The ADNET diagnostician for DNS configurations can diagnose configuration problems with DNS. The common DNS error message: `unknown host` may have different causes related to the various underlying configurations of the DNS. For each of these configurations, the diagnostician program constructed by ADNET figures out the problem and provides a more meaningful message. Before we give such an example, we briefly present the DNS configuration as it is required by BIND (the DNS implementation written for Berkeley's 4.3BSD UNIX), running on DEC OSF/1:

- The file `/etc/svc.conf` is consulted to see what services are available and in what order they are to be used, as indicated by the `hosts` statement in this file. The possibilities are `local` and `bind`. *Local resolution* is used for small networks configured by a single administrator, with no traffic to outside world, whereas *bind resolution* becomes a "must" if the local network is connected to a larger network.
- The *local resolution* is provided by consulting the file `/etc/hosts` which contains a table of known names and IP addresses.
- The *bind resolution* is provided by contacting a server daemon, called **named**. The client that accesses the name server is called a *resolver*, and its configuration is defined in the `/etc/resolv.conf` file. A resolver creates the query, sends it across a network to a name server, interprets the response, and returns it back to the program that requested it. If the `/etc/resolv.conf` file is present, it is consulted to find an ordered

list of the IP addresses of server daemons to be contacted. Resolution fails if none of the servers responds, or if the first one that does respond is not able to resolve the name. If the `/etc/resolv.conf` file is not present, an attempt is made to contact the *local server daemon*, running on the local host.

- Each name server has its own configuration, as a *primary server*, *secondary server*, or *forwarder*. While the resolver configuration requires, at most, one configuration file, several files are used to configure **named**. For example, the boot file defines the type of the server and the location of other configuration files, such as the database files, the loopback address file, and the cache data file.

Although the above specification of the DNS configuration is not complete, it can be used to completely model the local resolution service. The diagnosis shown in Figure 3 explains the simple problem caused by incompletely specifying the `/etc/hosts` file, when only the `local` option is specified in the `/etc/svc.conf` file. The repair procedure for this problem is straightforward once the cause of the problem is clearly explained. More subtle

Command: telnet alpha
DNS Error Message: unknown host: alpha
Configuration: **Local resolution only is indicated in** /etc/svc.conf. alpha **name is not in** /etc/hosts.
Diagnosis: ***** Local resolution failed. No** alpha **in** /etc/hosts.

Figure 3 DNS configuration diagnosis: incompletely specified host table

problems, such as forgetting to increment the serial number of the primary's zone files, or even syntax errors in the boot and database files, can be diagnosed by a diagnostician compiled with ADNET, if a complete DCSP model of the DNS configuration is provided.

6 ADNET MODELING LANGUAGE

Based on studies of FTP and DNS network services, and models built for them using the CSP formalism, we have designed a simple language for describing models in CSP terms. A model specified in the ADNET modeling language consists of four sections. The first two sections define (1) the set of *variables* and their corresponding *domains of values*, and (2) the set of *constraints*, also called *compatibility constraints*. The next two sections specify the *activation* of those variables and constraints which are relevant to the observed configuration decisions. Figure 4 presents that part of the local resolution model for the DNS example which describes the variables that may play a role in localizing the faults with the configuration of local resolution, and the constraints that restrict the values these variables can take.

The variables are defined with VAR statements. Each variable has a *name* and a *domain of possible values*. The modeling language offers two built-in mechanisms for specifying the domain of values for a variable. The DEF slot of the VAR statement specifies the domain of values known at modeling time, while the ASK slot represents the mechanism for supplying values at diagnosis time, either from the running network directly or from the user who observes the network. Referring to Figure 4, the `remote-host` variable has

the value returned by the ASK function `prompt-user`, while the variable `ping-path` has a single predefined value ``/sbin/ping'', specified in the DEF slot. ASK functions invoke system calls for internally managed information, such as currently running processes, or prompt the user. The ADNET system offers some built-in network probing tools, such as `prompt_user` and `ping`, and provides the means to add new functions, customized for specific measurements. User-defined ASK functions, such as `resolve-services` and `resolve-host` shown in Figure 4, are written as *generator functions*, allowing the user to investigate network resources and get possible values for the current variables, one value at a time. The function is called repeatedly to return all values in the domain, until the values are exhausted or an error condition is encountered. In the latter case, the error message reports the violation of the unary constraint implicitly associated with ASK-ed variables, and is added to the diagnostic messages of the current violated constraints.

```
VAR remote-host       ASK prompt-user(``Remote host name:'')
VAR ping-path         DEF ``/sbin/ping''
VAR ping-response     ASK ping($remote-host)
VAR services-file     DEF ``/etc/svc.conf''
VAR resolution-type   ASK resolve-services($services-file, ``hosts'')
VAR hosts-file        DEF ``/etc/hosts''
VAR hosts             ASK resolve-host($hosts-file)
CON $remote-host IN $hosts
    ``*** Local resolution failed. No $remote-host in $hosts-file.''
```

Figure 4 The VAR and CON declarations of the DNS local resolution model

The constraints are defined with CON statements. A constraint is defined on a *set of variables* and can be specified either extensionally, as the *set of tuples* of values allowed by the constraint, or intensionally, as *predicates*. The ADNET modeling language provides the standard logical, set, and relational operators. The operands can be constants (strings, numbers), current values of instantiated variables selected with *$variable-name*, or values returned by user-defined function calls. In addition, the user can supply his own predicates, taking as arguments any of the above. Part of the constraint declaration is a required *diagnostic message* issued in case the constraint is violated.

In general, the configuration task consists of assembling parts into a whole. Since the parts are taken from a larger (but fixed) set, some parts will never be used in the configured system. Moreover, the process of configuration reflects dependency relations as to how some parts of the configurable system require/exclude other parts to/from being present, unconditionally, or under specific conditions. The process of configuration starts with some key parts, always required in the system. To model these new features, the DCSP extensions to the standard CSP are:

- among all the CSP variables, some of them become *active variables*, as they are relevant to the configuration decisions checked at some point during the search,
- variable activity is controlled with the *activity constraints*,
- there is an initial set of active variables, called *start variables*.

A program written in the ADNET modeling language for diagnosing a configuration problem adds to the VAR and CON sections in Figure 4 two more sections:

- the START section, which defines the set of initially active variables, and
- a section that defines the set of activity constraints (*always require variable* and *require variable* constraints, introduced by the ARV and RV statements, respectively).

Figure 5 completes the DNS configuration model described in Figure 4 with the DCSP information, namely, the declaration of the START variables, and the activity constraints.

```
START remote-host
ARV   remote-host ⇒ (ping-path ping-response)
RV    $ping-response = ``unknown'' ⇒ (services-file resolution-type)
RV    $resolution-type = ``local'' ⇒ (hosts-file hosts)
```

Figure 5 The START variable and activity constraints of the DNS local resolution model

The START variable `remote-host` has to be instantiated no matter what configuration decisions are followed in the model. Once the value of this variable is known, two other variables become part of the diagnosis process: `ping-path` and `ping-response`. These variables are always required by the `remote-host` variable, regardless of its value, as the ARV constraint shows in Figure 5. The remaining activity constraints in the example are *require variable* constraints. These constraints activate certain variables based on the values assigned to the already active variables. For example, if the `ping-response` variable has the value "unknown", then the information about the service file `/etc/svc.conf` becomes part of the search space. Similarly, when the value "local" is observed for the `resolution-type` variable, only the host table information is further explored.

Assembling the CSP specification in Figures 4 and 5, we obtain the complete model for diagnosing the DNS local resolution. The declarative nature of the CSP formulation permits us to easily extend this model for diagnosing BIND resolution as well. To provide a basic insight on how further information about DNS configuration can be added to the local resolution model, we make use of a graphical representation of the ADNET modeling language constructs. Figure 6 illustrates the DNS local resolution model and outlines how the BIND resolution model can be built from the DNS problem description given in Section 5. The conventions in this graphical representation are simple:

- the circles, labeled CON, are the compatibility constraints, whose violations provide the diagnostic message associated with them.
- the arrows are the active variables, and are labeled with their names. These variables change dynamically in response to decisions made during the course of problem solving. An outward arrow shows the required activation of the variable, while an inward arrow shows the active variable requiring the activation.
- the variable activity is controlled by the activity constraints, drawn as boxes. They are either ARV constraints or RV constraints. In Figure 6, the ARV constraint in the Local Resolution Model activates the variables `ping-path` and `ping-response` once the variable `remote-host` is active. The RV constraints activate other variables if some already active variables satisfy some condition. Since the condition of the RV constraints

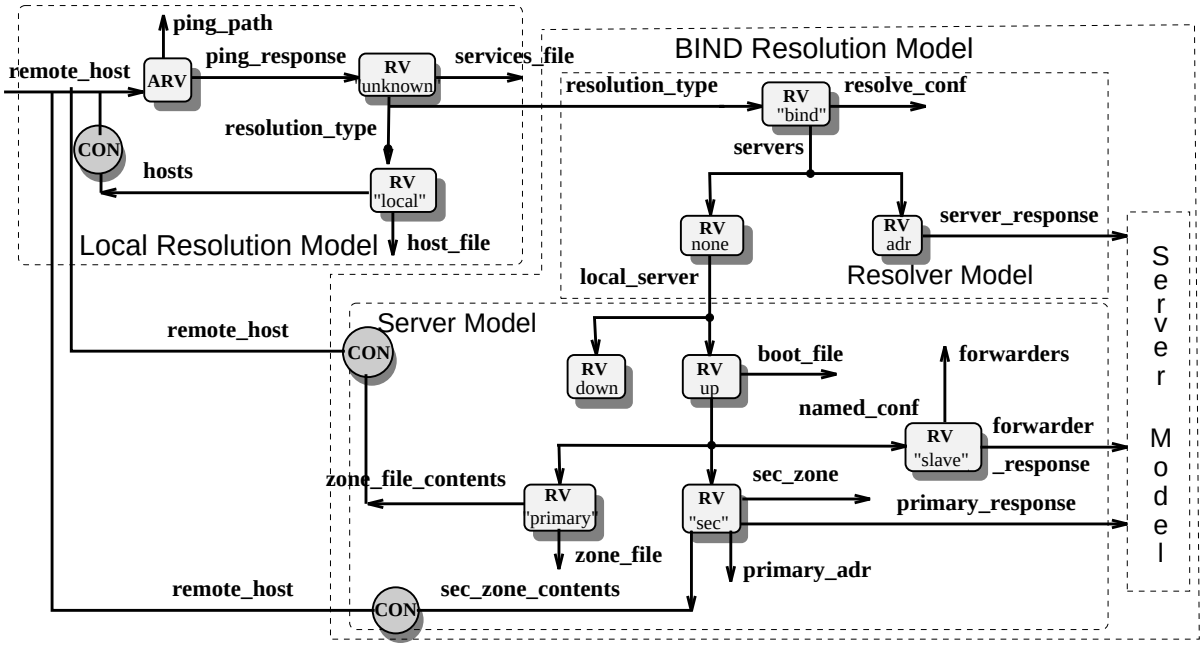

Figure 6 DNS configuration model formulated as a DCSP

in the example in Figure 6 requires some already active variable to have a specific value, the box representing the activity constraint is labeled with that particular value.

Although the BIND Resolution Model shown in Figure 6 is not completely specified, the description given is still a working model, in the sense that the diagnosis covers the configuration problems produced by the failure of any of the compatibility constraints specified in the model, and the failure of the consistency check of any implicit unary constraint associated with the access to the domain values of a variable.

7 RELATED WORK

Knowledge-based technologies are characterized by utilizing domain knowledge represented in a declarative form, and human expertise expressed as rules of inference applied to the domain knowledge for performing a specific task. Knowledge-based systems for network fault management evolved from rule-based reasoning (RBR) systems to case-based reasoning (CBR) systems, and, more recently, to model-based reasoning (MBR) systems. In the following, we briefly illustrate this line of evolution, and outline the limitations each approach has encountered, as well as the solutions that have been proposed either within each approach or, more radically, by another one.

The well-known problems inherent to the *RBR systems* are the brittleness problem, or the impossibility of coping with unforeseen situations, and the knowledge acquisition problem, which arises when knowledge base growth endangers the manageability and consistency of the knowledge base itself. These problems are not critical in small, homogeneous, relatively static networks, but cannot be ignored in today's telecommunications networks, with their high rate of technological change. Thus, RBR systems become un-

maintainable and unpredictable as more ad hoc rules are added to the knowledge base, with the imminent effect of proliferating unintended rule interactions and conflicts. However, different strategies of structuring the knowledge base hierarchically and providing higher-order relationships among constituent modules help diminish these problems, as is shown in the expert systems presented in (Frontini *et al.* 1991), (Schröder and Schödl 1991), (Lor 1993).

A *CBR system* addresses the brittleness and knowledge acquisition problems by exhibiting learning and adaptability capabilities. Past experience is accumulated and retrieved whenever identical or possibly similar situations are encountered. Unanticipated cases are solved by adapting existing ones. Once solved, the cases are "learned" by being added to the case repository. The prototype system for network traffic management described in (Goyal 1991) is one example of how a case-based reasoner can recognize, treat, and monitor traffic routing problems. Another example is presented in (Lewis 1993), where the diagnosis functionality is built on top of a trouble ticket system. The critical factors in CBR, however, are the similarity metrics based on which the retrieval of cases similar to the current one is possible, and the adaptation methods that ensure the transformation of the current case into one for which the solution is already known.

The *MBR paradigm* has emerged from the need to overcome the long-term, ever-growing dependency of the system on the experience gained with the system itself. The expert knowledge, which forms the empirical associations collected into rules in a RBR system, or the similarity metrics and adaptation functions in a CBR system, is now formalized into the *model* of the system to be managed. (Jordaan and Paterok 1993) describes a prototype event correlation application which needs "very little or no preconfigured knowledge", compared to RBR systems. The underlying idea is that in practice almost all objects modeling the network are related in some fashion, but just a few relationships prove to describe fault propagation effectively, and cover the vast majority of, otherwise ad-hoc, heuristic associations. Keeping modeling simple is the underlying idea in (Crawford *et al.* 1995), where the approach outlined in (Jordaan and Paterok 1993) is further formalized.

A distinct modeling technique that has recently emerged in the field of network management is the utilization of constraints. However, the constraint-based technique has not been explicitly related to the existing work on *constraint satisfaction problems* (CSPs) as they are well-known in the artificial intelligence community. For example, (Goli *et al.* 1995) proposes a constraint-based solution to the problem of checking MIB update validity, and describes the design of a network constraint management system to implement this approach. Another application of constraints is the modeling of temporal relations for the event correlation task, as presented in (Jakobson and Weissman 1995). The model captures temporal constraints and, thus, reasons about time. Constraints are also used to describe connectivity and containment relationships for the network configuration model. The description language presented in (Pell *et al.* 1995) is centered around the constraint concept. It supports network fault management through the means of checking all the constraints that define the characteristics of a particular network resource.

8 CONCLUSION

In this paper we proposed a constraint-based modeling and problem-solving approach to the diagnosis of network services. This approach is based on the concept of constraints

which capture the structure and behavior of the system to be diagnosed, and which represent relations among system components. The automatic diagnosis employs a diagnostic engine which outputs the expected diagnoses, based on the model of the system and the observations performed during system operation. We further automated this diagnostic scheme along two directions: (1) a modeling language is provided to describe, in a declarative way, the structure and behavior of the network service, and (2) observational data can be dynamically requested from the running network by incorporating general-purpose and user-defined probing tools in the diagnostic system. We showed how these features have been incorporated in ADNET, a prototype system which automates the construction of specialized C++ diagnostic tools. We also described the ADNET modeling language and illustrated its use in a model for diagnosing configuration problems with DNS. The ADNET architecture makes possible the integration of diagnostic tools within existing network management platforms, so that the collection and display of managed data can be complemented with the problem-solving capabilities of ADNET diagnosticians.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No. IRI-9504316, and by Digital Equipment Corporation, for which we would like to especially acknowledge the contributions of Neil Pundit and Ed Valcarce.

REFERENCES

Crawford, J., Dvorack, D.L., Litman, D., Mishra, A.K. and Patel-Schneider, P.F. (1995) Device representation and reasoning with affective relations. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1814-1820.

Freuder, E.C. and Wallace , R.W. (1992) Partial constraint satisfaction. *Artificial Intelligence*, **58**, 21-71.

Frontini, M., Griffin, J. and Towers,S. (1991) A knowledge-based system for fault localization in wide area networks. In I. Krishnan and W. Zimmer, editors, *Integrated Network Management, II*, 519-530. Elsevier Science Publishers B.V., North-Holland.

Goli, S.K., Haritsa, J. and Roussopoulos, N. (1995) Icon: A system for implementing constraints in object-based networks. In A.S. Sethi and Y. Raynaud, editors, *Integrated Network Management, IV*, 537-549. Chapman & Hall, London,.

Goyal, S.K. (1991) Knowledge technologies for evolving networks. In I. Krishnan and W. Zimmer, editors, *Integrated Network Management, II*, 439-461. Elsevier Science Publishers, B.V., North-Holland.

Hamscher, W., Consosle, L. and de Kleer, J., editors (1992) *Readings in Model-Based Diagnosis*. Morgan Kaufmann Publishers, San Mateo, CA.

Jakobson, G. and Weissman, M. (1995) Real-time telecommunication network management: extending event correlation with temporal constraints. In A. S. Sethi and Y. Raynaud, editors, *Integrated Network Management, IV*, 291-301. Chapman & Hall, London.

Jordaan, J.F. and Paterok, M.E. (1993) Event correlation in heterogeneous networks using the OSI management framework. In H.-G. Hegering and Y. Yemini, editors, *Integrated Network Management, III*, 683-695. Elsevier Science Publishers B.V., North-Holland.

Lewis, L. (1993) A case-based reasoning approach to the resolution of faults in communication networks. In *Integrated Network Management, III*, 671-682. Elsevier Science Publishers B.V., Amsterdam.

Lor, K.-W. E. (1993) A network diagnostic expert system for Acculink multiplexers based on a general diagnostic scheme. In H.-G. Hegering and Y. Yemini, editors, *Integrated Network Management*, 659-669. Elsevier Science Publishers, B.V., North-Holland.

Meyer, K., Erlinger, M., Betser, J., Sunshine, C., Goldszmidt, G. and Y. Yemini (1995) Decentralizing control and intelligence in network management. In A.S. Sethi and Y. Raynaud, editors, *Integrated Network Management, IV*, 5-15. Chapman & Hall, London.

Mittal, S. and Falkenhainer, B. (1990) Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, 25-32.

Pell, A.R., Eshgi, K., Moreau, J.-J. and Towers, S.T. (1995) Managing in a distributed world. In A.S. Sethi and Y. Raynaud, editors, *Integrated Network Management, IV*, 95-105. Chapman & Hall, London.

Rose, M.T. (1993) Challenges in network management. *IEEE Network*, 7(6), 16-19.

Sabin, D., Sabin, M., Russell, R.D. and Freuder, E.C. (1995) A constraint-based approach to diagnosing software problems in computer networks. In *Proceedings of the 1st International Conference of Principles and Practice on Constraint Programming*,.

Schröder, J. and Schödl, W. (1991) A modular knowledge base for local area network diagnosis. In I. Krishnan and W. Zimmer, editors, *Integrated Network Management, II*, 493-503. Elsevier Science Publishers, B.V., North-Holland.

BIOGRAPHIES

Mihaela Sabin received her MS in Computer Science from the Polytechnic Institute of Bucharest, Romania, in 1984. Currently, she is working towards her PhD in Computer Science at the University of New Hampshire. Her research interests include constraint satisfaction, diagnosis, modeling, and network management. She is a student member of AAAI and IEEE. Her home page address is http://www.cs.unh.edu/~mcs.

Robert D. Russell is an associate professor in the University of New Hampshire Department of Computer Science. His research interests include network protocol development, LAN-based parallel programming, ATM Quality of Service specification, and network management. He is a member of IEEE and ACM.

Eugene C. Freuder is a professor in the University of New Hampshire Department of Computer Science and Director of its Constraint Computation Center. He is a Fellow of the American Association for Artificial Intelligence. He is the founding editor of *Constraints*, An International Journal (Kluwer Academic Publishers) and a member of the Organizing Committee of the International Conference on Principles and Practice of Constraint Programming. His home page address is: http://www.cs.unh.edu/ecf.html.