Empowering Self-diagnosis with Self-modeling

Carole Hounkonnou, Eric Fabre
INRIA Rennes Bretagne Atlantique
Email:{carole.hounkonnou, eric.fabre}@inria.fr

***Abstract*—This paper proposes an approach to automatise the management of faults, covering the different segments of a network, and the end-to-end services deployed over them. This is model-based approach addressing the two weaknesses of model-based diagnosis namely deriving an accurate model and dealing with huge models. To address the first point, we propose a solution called self-modeling that formulates off-line generic patterns of the model, and identifies on-line the instances of these patterns that are deployed in the managed network. The second point is addressed by an active (self-)diagnosis engine, based on a Bayesian network formalism. This consists in reasoning on a progressively growing fragment of the network model: more observations are collected and new tests are performed until the faults are localized with sufficient confidence. This active diagnosis approach is experimented to perform cross-layer and cross-segment alarm management on an IMS network.**

***Index Terms*—Self-modeling, Self-diagnosis, Fault localization, Alarm correlation, Bayesian networks, IMS**

I. Introduction

While modern networks and services are continuously growing in scale, complexity and heterogeneity, the management of such systems is reaching the limits of human capabilities. Technically and economically, more automation of the classical management tasks is needed. This has triggered a significant research effort, gathered under the terms 'self-management' and 'autonomic networking'.

One important management task is the processing of faults and alarms [1]. Traditionally limited to alarm filtering and simple rule-based event correlation, this task moves towards large-scale, cross-layer and cross-segment fault monitoring.

Many approaches to fault management rely on black-box or learning methods ([2], [3], [4]–[7]). However, their shortcomings make them appropriate for simple and small scale event correlation over fixed topologies, where training can safely converge. To progress towards the more ambitious objectives mentioned above, one must adopt a model-based strategy.

Several contributions adopt this strategy. Some of them compile expert knowledge about malfunctions and their consequences, or symptoms and their causes, into causal graphs [8], [9] and [10]. Others model or learn from data the dependencies between resources, initial faults and observable symptoms, and compile them into Bayesian networks or related objects [11]–[14]. The inference engine on such structures is a well covered topic, and the distribution of these techniques is also a well paved trail [14], [15]. However, model-based techniques always raise two major difficulties: a) how to derive such a model, suited to a given network at a given time, especially to capture several network layers and segments (see open problems 1, 4 and 6 in [1]) and b) how to reason on a potentially huge model, especially to manage a nation-wide network for example. This paper proposes a contribution to these two issues.

Obtaining a model of the managed network is far from simple, particularly to capture inter-layer and inter-segment fault propagations. The proposed self-modeling approach consists first in identifying the different families of managed network resources, and how these resources are structured and rely on one another. This yields a collection of generic patterns of resources and their dependencies, designed according to a specific grammar. Secondly, by exploring the managed network, one can then create as many instances of these generic patterns as discovered in the network topology. As these instances share some resources, this construction results in a large scale structure where similar patterns are duplicated and partially overlap. The model obtained in that way perfectly matches a given network, and can be used for diagnosis, and possibly for failure impact analysis.

Regarding the diagnosis engine, we propose to translate the model of network resources and their dependencies into the Bayesian network (BN) formalism, which seems to reach some consensus in the network management community. But dealing with possibly huge models may make Bayes nets inappropriate. Therefore we propose to adapt the BN formalism to explore only part of the model, starting by the resources involved in a given malfunction to be explained, and progressively introducing/revealing more resources (*i.e.* variables) to reach new observations and making new tests, in order to locate the origin of the malfunction with more precision.

To support these two research directions, we consider the management of failures in IMS networks, capturing several network segments, and the end-to-end services running on top of them [16]. Section II shows how this use-case structures the nature of models that should be used for cross-layer and cross-domain fault correlation. Section III illustrates on a simple example the typical reasoning that can be performed on a model obtained by self-modeling. Section IV formalizes this approach as a progressive inference problem on a generic Bayesian network, a possibly large BN made of many overlapping similar patterns of variables. A set of experiments demonstrates how the algorithm progressively explores the model, picking the most informative observations, in order to locate a failure. Finally, the concluding section explains how we plan to pursue this work towards a true experiment of IMS network management, and give hints on the extension of this work towards failure impact analysis and healing suggestions.

978-3-901882-48-7

II. Structure of IMS networks

A. Four multiresolution layers

Starting from standard descriptions of IMS networks, the objective here is to identify the network resources involved in such networks, their structuring and their dependencies, which will be the knowledge used to diagnose malfunctions.

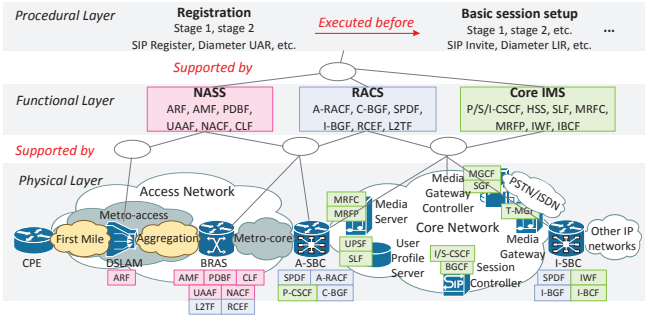

Fig. 1. IMS networks are hierarchically organized structures.

Figure 1 illustrates the generic architecture of an IMS network, which can be organized into four layers (three displayed), each one gathering resources of a specific nature. The term 'resource' covers both physical equipment in charge of transport, pieces of software running over them, but also procedures, and finally high level capabilities. The physical layer-comprises an access network and a core network. The access network is made of metro-access and metro-core segments. The metro-access decomposes into smaller segments: first mile and aggregation. Aggregation and metro-core segments are connected via one or more BRAS. The metro-core segment is connected to the core network, containing the IMS service platform, via routers such as the Session Border Controller (SBC). The physical layer architecture is essentially hierarchical: it is made of specific equipment connecting various network segments, the latter being decomposable into other physical equipment connecting smaller network segments, etc.

The functional layer refers to the TISPAN NGN IMS functional architecture [17] and is made of three main subsystems. The Network Attachment Subsystem (NASS) [18] provides registration at the access level and initialization of the User Equipment (UE) for accessing to the multimedia services. The Resource and Admission Control Subsystem (RACS) [19] is in charge of policy control. Finally the Core IP Multimedia Subsystem [20] (core IMS) is in charge of session initiation, provisioning and termination for multimedia services directed to terminal users. Subsystems are hierarchical objects, comprising several functions (listed within each main box of the functional layer in Fig. 1) that communicate with each other via interfaces (also called reference points). Various solutions exist to map these functions to physical equipment (or nodes) and the authors in [21] compare different implementations. Fig. 1 reflects a possible implementation with an xDSL-based access network. This embedding is modeled by a 'supported by' relation between a function and a physical equipment. This is a first example of resource dependency since the failure of an equipment generally impacts the functions running over it.

The procedural layer describes the procedures involved in the normal operations granting access to IMS services. Such procedures, described in standards, have a multi-resolution structure: they decompose into stages or phases, that themselves decompose into sequences (or partial orders) of request/answer between functional elements of the NASS, RACS and Core IMS (see Fig. 3 for an example). Each such request/answer decomposes further into exchanges over interfaces that obey standard protocols, such as DHCP, SIP or Diameter. The execution of a procedure positions state variables, some of which are observable or can be tested. For example, the correct getting of an IP address proves that the network attachment of the User Equipment (via the NASS) was correctly performed (see Fig. 3). Procedures, phases, and their smaller elements depend on each other, in the sense that they are (partially) ordered in time, which we denote by an 'executed before' relation. But they are also 'supported by' the functions and interfaces of the functional layer that run these procedures. This information is easily accessible in the standards, and reveals another form of dependency between resources.

A fourth layer, not illustrated in Fig. 1, describes more complex service 'capabilities' that are obtained by assembling procedures in an appropriate manner. Examples are the ability to establish a connection with an AS, the ability to make calls from a city to another one, or within the same city, the ability to call a user located in the IMS network of another operator, etc. Such capabilities provide high-level indicators of the network state. They can be both tested (e.g. in view of failure impact analysis), and used as an entry point (*i.e.* as a model) to enable a diagnosis when a user complains about some service. These network capabilities are only partly derivable from the standards, and should mostly reflect the main functions an operator wishes to check and diagnose in its network.

B. Generic model vs network instance: self-modeling

The four-layer model described above is *generic* since it defines the different *types* of network resources involved, how they interact and depend on each other. Most of it derives from the standards or best practices of an operator, but part of it must be designed by an expert. This work is facilitated by the relatively small size of this generic model, by the existence of a grammar defining how objects can be related, and by the hierarchical nature of the model. This allows one to model each layer by progressive refinements, thus selecting the finest granularity at which one wishes to manage the network.

The generic model defines the building blocks of a network, from a management perspective, but the actual network one has to manage contains many *instances* of these elements. Borrowing to object oriented programming, the generic model is a class diagram, while one has to monitor an object diagram. The actual network can be represented as a large collection of instances of the patterns described in the generic model, and these instances overlap on some common resources. This is

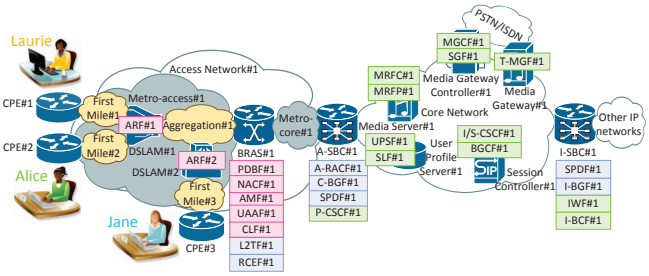

Fig. 2. An IMS network instance: physical layer, with mention of the supported functions of the functional layer.

illustrated in Fig. 2, where several users of an access network have private parts (CPE and first mile), but may or not share a DSLAM, may or not be connected to the same aggregation segment, etc.

Deriving the model of the actual network instance to be managed means creating as many instances as necessary of the network resources, as they are described in the generic model. It also means structuring or connecting them in accordance with the patterns allowed by the generic model. Such a construction guarantees both the adequacy of the resulting model with the standards, and allows one to fit a model to a specific network, which is of great interest to track evolving architectures. We call this process the *self-modeling*. This task can indeed be automatised provided the management framework provides tools to explore the network and reveal its architecture. Such a 'service,' or this 'reflectivity property' of the network, is one of the essential functions one should expect from a self-management framework, together with the ability to query the managed elements in order to check some of their state variables or to perform tests.

III. Self-modeling as support for fault localization

A. Methodology

The complex dependency relations illustrated in the previous section naturally orient us to a formalization in terms of Bayesian networks (BN), which are particularly suited to represent both constraints and statistical dependencies.

The framework needs adaptations however, in several directions. Firstly, the managed network evolves in time and may not be known entirely and in full details. So the BN used for inference should be built on request, capturing the state of the network, to answer a given diagnosis query. Secondly, not all network resources are involved in the malfunction of say some high level capability. So only part of the network should be taken into account. Thirdly, this BN model construction should be coupled with the inference engine. Users share some of the network resources, therefore they carry information about their state, which can be useful to the reasoning. So one must design a dynamic construction of the BN modeling the network, or equivalently a dynamic exploration of the network, to collect more information and solve a given diagnosis query. Finally, let us clarify what this diagnosis query means. Assuming some network resource is observed as down (e.g. IP configuration down for a specific UE), one has to discover the root cause of this failure, which necessarily lies in the lower level resources that are assembled to build the damaged one. This can be understood as a state inference problem given the values observed on other state variables. By extension, one could imagine querying the state of *all* variables of a given *type*, given some observations. This corresponds to a questions like, for example, given access segment failure, what is the probability that a *non specified UE* be not able to make calls. This type of generic inference query is new in the BN formalism, and is clearly an approach to perform fault impact analysis.

Our approach to troubleshoot an observed malfunctioning resource is based on the following methodology:

1) Find and/or assemble (at some level of abstraction) the generic model (or generic BN) that describes the resources used by the malfunctioning resource.
2) Locate the instance of this generic BN in the IMS network instance, thus deriving a BN instance, that we call a *pattern*.
3) Within the current BN (instance), feed the inference engine with available observations to locate the faulty resource.
4) If the collected observations are not sufficient, expand the current BN by exploring other patterns (BN instances) that share resources with the current BN. In this expanded BN, collect the new available observations to improve fault location.
5) Repeat the expansion until confidence in the explanation becomes sufficient.

B. Example

To illustrate this methodology on a simple practical case, suppose that we want to explain why the IP configuration failed for the user Laurie in the network instance of Figure 2. As shown by Figure 3, the IP configuration capability is broken down into two successive stages, initiated by the UE.

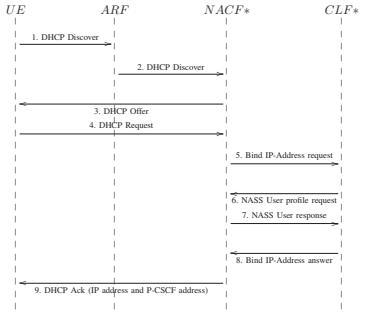

Fig. 3. Sequence diagram of IP configuration using DHCP (procedural layer). It consists in a dialog between the User Equipment (left) and functions of the NASS (right).

In the functional layer, the chosen granularity of description distinguishes a single resource denoted 'NACF*' that groups the AMF, the NACF, the UAAF, the communication interface

between AMF and NACF (int AMF-NACF) and the one between NACF and UAAF (int NACF-UAAF). In the same manner, the chosen granularity distinguishes a single resource denoted 'CLF*' that groups the CLF, the A-RACF and the communication interface (int CLF-A-RACF) between these two functions. Details about the information flows displayed in Figure 3 and their meaning can be found in [18].

Following the methodology described above, we first retrieve the generic model that describes the resources used by the IP configuration capability (Fig. 4). This dependency graph between resources encodes, for example, that the outcome of the 'IP configuration' procedure demands that 'Stage 1' be properly performed, which depends on the state of the interface between the functions UE and ARF denoted here as 'int UE-ARF', which in turn uses the communication channel between the UE and the ARF, that is the first mile segment. In this generic BN, the state variable 'IP configuration' is observable, since one can easily test whether the UE obtained a correct IP address.

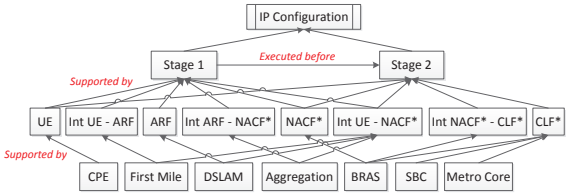

Fig. 4. Generic BN for the IP configuration capability of a User Equipment.

As a second step, we locate among multiple instances of this generic BN, the instance that refers to the user Laurie. This BN instance is shown in Figure 5 where Laurie's private resources are displayed in orange. The state of the observable variable 'IP configuration' in the BN instance attached to Laurie is set to 'down,' *i.e.* not functioning correctly.

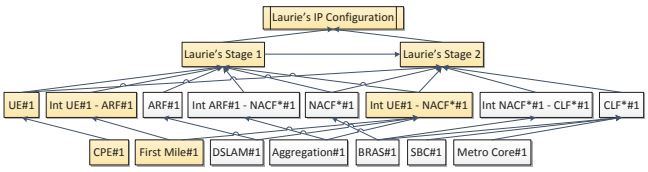

Fig. 5. Laurie's BN instance

Any node in Laurie's BN instance can be an explanation for the observed malfunction. The third step consists in collecting the available observations on all these resources, and running a BN inference, to try and locate the origin of the malfunction. Note that not all resources may provide observable state information and that some observations may not be fully discriminant.

As a fourth step, one may decide to cross-check the possible explanations discovered so far by querying other users that share some resources with Laurie. For example, we choose check the state of Jane's IP configuration. Thus, we extend the scope of the BN under study (Laurie's BN instance) by adding Jane's BN instance. Figure 6 shows the extended BN, which now incorporate more observable variables. Suppose that the state of Jane's IP configuration is 'up,' functioning correctly. This observation implies that all the resources Laurie shares with Jane are working properly (which would reveal a BN inference on this extended BN). As a consequence, the set of possible faulty resources in Laurie's BN instance is reduced to the subset of resources that are not shared with Jane.

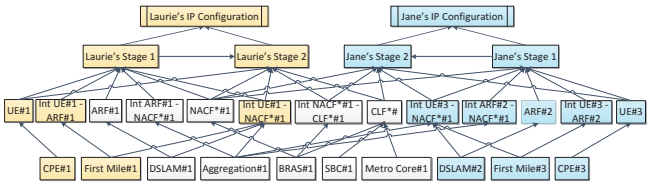

Fig. 6. Jane's BN instance is added; her private resources are represented in blue.

This process can be iterated (point 5): one may query another user, provided he/she shares resources with either Laurie or Jane. But the choice of the user to query should depend on how informative the new observations will be to locate the faulty resource that caused Laurie's problem. If we choose to query Alice, who has more resources in common with Laurie than Jane has, the current BN must again be extended to incorporate Alice's BN instance of the IP configuration generic model. Figure 7 shows the extended BN (due to space limitation Jane's private resources are not displayed). Suppose that Alice's IP configuration is 'up.' This observation implies that all the resources Laurie shares with Alice are functioning. Once more, the set of possible faulty resources in Laurie's BN instance is reduced to the subset of resources that are not shared with Alice or Jane. By contrast, suppose that Alice's IP configuration is 'down.' This observation increases our confidence in the state 'down' for the resources in Laurie's BN instance that are shared with Alice, and not shared with Jane. The cause of Laurie's IP configuration failure is likely to be found among these shared resources.

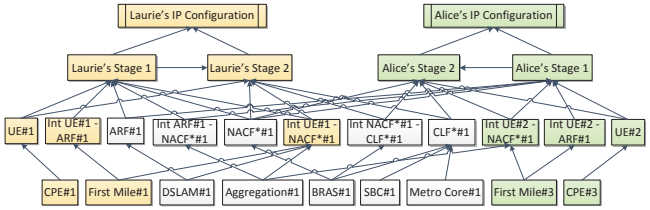

Fig. 7. Alice's BN instance is added; her private resources are represented in green. Jane's private resources are not shown.

IV. Reasoning with Generic Bayesian Networks

The previous sections demonstrated how the network resources depend on each other, forming a huge dependency graph. This graph can be modeled as a Bayesian network where many parts are isomorphic. We formalize here its construction and explains how to perform inference over it.

A. Formalization

Definition 1. A *Bayesian network* (BN) $X = (V, E, \mathbb{P}_X)$ is formed of a finite DAG $G = (V, E)$, with V as vertex set and $E \subseteq V \times V$ as edge set, and a collection of random variables $(X_v)_{v \in V}$ indexed by V and with probability distribution $\mathbb{P}_X$. Denoting ${}^\bullet v = \{u \in V : (u, v) \in E\}$ the parents of node $v \in V$ in the DAG (V, E), the distribution of X factorizes as $\mathbb{P}_X = \otimes_{v \in V} \mathbb{P}_{X_v | X_{{}^\bullet v}}$, where $X_U = (X_u, u \in U)$ denotes a vector of random variables, $U \subseteq V$.

As usual, a *morphim* $\phi : G_1 \to G_2$ between DAGs $G_i = (V_i, E_i)$ is a partial function from V_1 to V_2 preserving the edges: $\forall u, v \in Dom(\phi) = V_1' \subseteq V_1$, $(u, v) \in E_1 \Leftrightarrow (\phi(u), \phi(v)) \in E_2$. ϕ is said to be an *insertion* of G_2 into G_1 iff ϕ restricted to its domain is bijective, *i.e.* $\phi_{|V_1'}$ is an isomorphism between $G_{1|V_1'} = (V_1', E_1 \cap V_1' \times V_1')$ and G_2.

Definition 2. A *Generic Bayesian network* (GBN) is a finite collection of ordinary BNs $(W^k)_{1 \leq k \leq K}$, where each BN $W^k = (V_k, E_k, \mathbb{P}_{W^k})$ is also called a *pattern*. An *instance* $X = (V, E, \mathbb{P}_X, (\phi_{k,i})_{k \leq K,\, i \in I_k})$ of this GBN is a standard BN $(V, E, \mathbb{P}_X)$ where

1) each $\phi_{k,i}$ is an insertion of pattern (V_k, E_k) into (V, E), and the DAG (V, E) is covered by such pattern instances: $V = \cup_{k,i} Dom(\phi_{k,i})$, $\forall (u, v) \in E$, $\exists \phi_{k,i} : u, v \in Dom(\phi_{k,i})$.
2) probability $\mathbb{P}_X$ is inherited from patterns W_k through the insertion morphisms $(\phi_{k,i})_{k \leq K,\, i \in I_k}$: $\forall v \in V$, one has
   a) if ${}^\bullet v = \emptyset$, then $\forall k, i : \phi_{k,i}(v) = u \;\Rightarrow\; \mathbb{P}_{X_v} \equiv \mathbb{P}_{W^k_u}$
   b) if ${}^\bullet v \neq \emptyset$, then $\forall k, i : \phi_{k,i}(v) = u, {}^\bullet v \cap Dom(\phi_{k,i}) \neq \emptyset \;\Rightarrow\; {}^\bullet v \subseteq Dom(\phi_{k,i})$, $\mathbb{P}_{X_v | X_{{}^\bullet v}} \equiv \mathbb{P}_{W^k_u | W^k_{{}^\bullet u}}$

In other words, to build X one takes many copies of the different patterns W^k, and aggregates them by sharing some of the random variables. To ensure consistency of this construction, each variable X_v appearing in several pattern instances must be defined by the same conditional probability. Fig. 8 gives an example of a GBN with a single pattern of five variables, and an instance of this GBN with six copies of the pattern, overlapping in different ways. Observe that a given variable may play different 'roles' according to the pattern instance where it is considered.

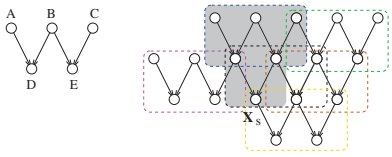

Fig. 8. A GBN (left) with a single pattern, and an instance of this GBN (right) with six overlapping instances of this pattern.

Fault localization in a GBN instance can be translated into a standard BN inference as follows. Assume some variable, for example X_s in Fig. 8, is declared/observed as faulty. This resource X_s depends on other resources, namely $X_{{}^{\bullet\bullet}s}$ where ${}^{\bullet\bullet}s$ denotes all the ancestors of node s. Either the failure of X_s is spontaneous, or it results from a fault propagation pattern in $X_{{}^{\bullet\bullet}s}$. The objective is thus to estimate the value of X_Z, with $Z = {}^{\bullet\bullet}s \cup \{s\}$ (in gray in Fig. 8). Alternatively, one can build a root-cause localizer L, as a function of X_Z that constrains the presence of a single root-cause within X_Z. The objective is then to estimate the value of L.

There exist observable variables within (or directly attached to) X_Z, in particular the observation indicating the faulty value of X_s. Let us denote by $Y_0 = y_0$ this observed vector. The conditional law $\mathbb{P}_{X_Z | Y_0 = y_0}$ gives a first information of what caused the failure of X_s: by maximum likelihood estimation, one gets an explanation (*i.e.* a fault propagation scheme) as the value $\hat{X}_Z = \arg\max_x \mathbb{P}_{X_Z | Y_0 = y_0}(x)$. How reliable this explanation is can be determined by computing the conditional entropy of X_Z given $Y_0 = y_0$:

$$H(X_Z | Y_0 = y_0) = \sum_x -\mathbb{P}_{X_Z | Y_0 = y_0}(x) \cdot \log_2 \mathbb{P}_{X_Z | Y_0 = y_0}(x)$$

A large conditional entropy means that the observations $Y_0 = y_0$ are insufficient to discriminate among several explanations. One may therefore seek to collect more observations in order to increase more information about X_Z, that is reduce the conditional entropy: the closer to zero, the most reliable the estimation of X_Z. The idea is the to explore a larger area of the BN instance in order to collect more observations, just like in the example of the previous section. From the explored set of nodes $U_0 = Z$ one thus goes to $U_1 \supseteq U_0$ such that U_1 is closed for the ancestor relation: ${}^\bullet U_1 \subseteq U_1$. This defines a new set of observations $Y_1 = y_1$, from which $\hat{X}_Z$ can be estimated provided $H(X_Z | Y_0 = y_0, Y_1 = y_1)$ is small enough. Otherwise one proceeds to another extension.

Referring to Fig. 8, one may extend the explored part of the BN instance to capture observations in either the green pattern instance, or those in the magenta one. Let us denote Y_1 and Y_2 these two possible sets of observations. The most informative one is obtained by comparing $H(X_Z | Y_0 = y_0, Y_1)$ to $H(X_Z | Y_0 = y_0, Y_2)$: the smaller one wins, which selects the set of observations that is the most informative *on the average*. This is computed beforehand, without querying/testing the actual value of the selected Y_i. Assume Y_1 was the most promising set of measurements, this defines the new area U_1 taken into account in the BN instance, where one may collect the observed value $Y_1 = y_1$. After standard inference, this yields the new posterior distribution $\mathbb{P}_{X_Z | Y_0 = y_0, Y_1 = y_1}$ and thus the conditional entropy $H(X_Z | Y_0 = y_0, Y_1 = y_1)$. Notice that $H(X_Z | Y_0 = y_0, Y_1 = y_1)$ may actually be either smaller or larger than the averaged value $H(X_Z | Y_0 = y_0, Y_1)$ used to determine the most promising set of observations.

The introduction of new measurements proceeds until either the conditional entropy is small enough, or no more observations are available. Notice that by construction of BN, the exploration is necessarily performed by exploring pattern instances that share variables with the explored section of the BN.

B. Experimental results

To demonstrate the relevance of the above exploration strategy, we have performed tests on a large tree-shaped GBN

instance. The (unique) pattern corresponds to Fig. 8 where variables X_D and X_E are observable. The instance is depicted in Fig. 9, where each vertex represents a pattern instance (thus 5 variables). Two connected pattern instances overlap by 1, 2 or 3 of the $\{A, B, C\}$ nodes, which is reflected by the thickness of the edge relating these patterns. In pattern (instance) X_1, variables $X_{1,A}, X_{1,B}, X_{1,C}$ are (independent) uniform binary variables. In all other patterns X_i, $i > 1$, the newly created variables $X_{i,A}, X_{i,B}$ or $X_{i,C}$ have distribution $(p, 1-p)$ with $p = 0.9$, in order to guarantee long-range correlation between patterns in the net of $Fig.$ 9. In pattern X_i, the observations Y_i correspond to variables $X_{i,D}, X_{i,E}$. D is more sensitive to a failure of B than to a failure of A, and similarly E reacts more to C than to B. Both sensors have a misdetection rate and a false alarm rate of 5%.

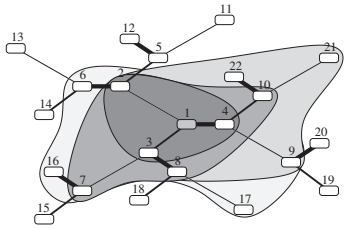

Fig. 9. A GBN instance with 22 pattern instances. The thickness (and length) of the line relating two patterns reflects the number of shared variables. To estimate the central pattern X_1, observations are introduced by zones, starting by the dark zones first and progressing towards the pale ones.

The experiment consists in drawing a random sample of the process described by this GBN instance, and in computing the conditional distribution $\mathbb{P}_{X_1|Y_U=y_U}$ of the central pattern given a growing set U of measurements. The measurement set starts from $U = \{1\}$ up to $U = \{1, 2, ..., 22\}$. In between, the most informative measurement is incorporated at each step. For each sample, the evolution of $H(X_1|Y_U = y_U)$ was computed. The experiment was conducted one thousand times. On the average, one observes that $H(X_1|Y_U = y_U)$ does decrease (Fig. 10), and quickly converges. However, for a given sample, this curve may not always decrease, as mentioned above: even if an observable node is very informative on the average, its actual observed sample may lead to a revision of the current assumptions, thus increasing temporarily the uncertainty.

More interestingly, we have performed rank statistics on the way measurements are introduced. On the average, the best order appears to be 1, 3, 4, 2, 7, 10, 8, 9, 21, 22, 17, 6, 16, 5, 13, 15, 20, 18, 19, 12, 14, 11, which is reflected by the growing zones around X_1 in Fig. 9. Strongly coupled patterns tend to be favored, but not always.

V. Conclusion

Model-based techniques are the key to an ambitious and autonomous management of network malfunctions. They are adaptable to network instances, offer accurate and wide range reasoning techniques, suggest methods to analyse the impact of failures and can go as far as suggesting the best mitigation actions. Their Achilles' heel is the derivation of an accurate

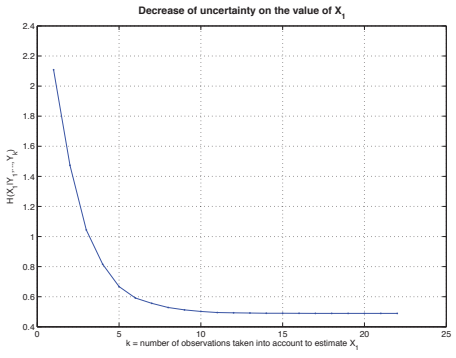

Fig. 10. Average evolution of the conditional entropy of the central pattern X_1 as the number of observations increases.

model of the managed network. We proposed a self-modeling principle that limits the burden of model construction to the design of generic patterns of the model, exploiting knowledge available in the standards. The actual model matching a given network instance is then built by connecting as many copies/instances as necessary of these generic patterns, in order to reproduce the dependency structure between the resources deployed in this network instance.

We proposed a formalism based on a notion of generic Bayesian network: a BN composed of many copies of a few patterns. Even if the model instance is large, one only needs to explore part of it to explain/diagnose some local observation, since variables located far away carry little information about the observed malfunction. 'How much information' can actually be fully determined and inserted in the reasoning. This formalism is not yet complete however, since it does not yet capture the intrinsic hierarchical construction of networks, that is the fact that a network segment generally decomposes into a structure of smaller sub-segments, and recursively. Such generic and multi-resolution BN do not exist yet, but seem crucial to network management. Other new and exciting research directions are opened by this approach, for example the simultaneous or successive processing of multiple diagnosis requests. Another one is the distribution of the reasoning to capture the fact that network segments are generally managed by different operational units. Finally, impact analysis can be performed by checking the influence of a given malfunction of all variables of a given *type* in a given generic pattern of the BN.

The next step of this work will of course be to demonstrate the relevance of this approach on typical failure scenarios in a realistic IMS architecture.

Acknowledgment

This work is supported by Alcatel-Lucent Bell Labs France, through the joint research team High Manageability of ALBLF and INRIA. It is also supported by the European Community through the UniverSelf project (FP7-IP). The authors would also like to thank Orange Labs for fruitful discussions about the IMS diagnosis use-case.

REFERENCES

[1] M. Steinder and S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53(2), pp. 165–194, 2004.

[2] S. Kavulya, K. Joshi, M. Hiltunen, and S. Daniels, "Draco: Top-down statistical diagnosis of large-scale voip networks," Carnegie Mellon University, Parallel Data Lab., Report CMU-PDL-11-109, Tech. Rep., April 2011.

[3] G. Reali and L. Monacelli, "Definition and performance evaluation of a fault localization technique for an ngn ims network," *IEEE Transactions on Network and Service Management*, vol. 6(2), pp. 122–136, 2009.

[4] C. Dousson, P. Gaborit, and M. Ghallab, "Situation recognition: Representation and algorithms," in *13th International Joint Conference on Artificial Intelligence, IJCAI,*, 1993.

[5] C. Dousson, "Alarm driven supervision for telecommunication network: on-line chronicle recognition," *Annales des Telecommunications*, vol. 51(9-10), pp. 501–508, Sept./Oct. 1996.

[6] C. Dousson and T. Vu Du'o'ng, "Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems," in *16th International Joint Conference on Artificial Intelligence, IJCAI,*, 1999, pp. 620–626.

[7] M. Kavulya, K. Joshi, M. Hiltunen, S. Daniels, R. Gandhi, and P. Narasimhan, "Practical experiences with chronics discovery in large telecommunications systems," in *SLAML '11, Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*, 2011.

[8] J. Lu, C. Dousson, and F. Krief, "A self-diagnosis algorithm based on causal graphs," in *ICAS'11: 7th International Conference on Autonomic and Autonomous Systems*, 2011.

[9] J. Lu, C. Dousson, B. Radier, and F. Krief, "Towards an autonomic network architecture for self-healing in telecommunications networks," in *AIMS'10: 4th Int. Conf. on Autonomous Infrastructure Management and Security*, 2010.

[10] I. Grosclaude, "Une approche a base de modeles pour les services de depannage (model-based problem solving for help and support services)," in *RFIA'08: French Conf. on Artificial Intelligence*, 2008.

[11] L. Bennacer, L. Ciavaglia, A. Chibani, Y. Amirat, and M. Abdelhamid, "Optimization of fault diagnosis based on the combination of bayesian networks and case based reasoning," in *IEEE/IFIP NOMS, Network Operations and Management Symposium*, April 2012.

[12] A. Bouloutas, S. Calo, and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Trans. on Communications*, vol. 42(2/3/4), pp. 523–533, Feb./March/April 1994.

[13] A. Bouloutas, S. Calo, A. Finkel, and I. Katzela, "Distributed fault identification in telecommunication networks," *Journal of Network and System Management*, vol. 3(3), pp. 295–312, 1995.

[14] E. Fabre, A. Benveniste, S. Haar, C. Jard, and A. Aghasaryan, "Algorithms for distributed fault management in telecommunications networks," in *ICT'04, Int. Conf. on Telecommunications*, 2004.

[15] E. Fabre, A. Benveniste, S. Haar, and C. Jard, "Distributed monitoring of concurrent and asynchronous systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 15, pp. 33–84, 2005.

[16] E. Bertin, I. Ben Yahia, and N. Crespi, "Modeling ims services," *Journal of Mobile Multimedia*, vol. 3(2), pp. 150–167, 2007.

[17] ETSI, "Telecommunications and internet converged services and protocols for advanced networking (tispan); ngn functional architecture." sep 2009.

[18] ——, "Telecommunications and internet converged services and protocols for advanced networking (tispan); ngn functional architecture; network attachment sub-system (nass)," mar 2010.

[19] ——, "Telecommunications and internet converged services and protocols for advanced networking (tispan); resource and admission control sub-system (racs); functional architecture." jun 2006.

[20] ——, "Digital cellular telecommunications system (phase 2+); universal mobile telecommunications system (umts); telecommunications and internet converged services and protocols for advanced networking (tispan); ip multimedia subsystem (ims); functional architecture." dec 2007.

[21] A. Darvishan, H. Yeganeh, K. Bamasian, and P. Eghtedari, "A practical ngn model by evaluation of various ngn solutions and its conformance with ims-tispan," in *Ubiquitous Information Technologies Applications, 2009. ICUT '09. Proceedings of the 4th International Conference on*, dec. 2009.