Probabilistic Reasoning for Fault Management on XUNET

Jean-François Huard
Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University, New York, NY 10027-6699

ABSTRACT

This document describes a fault management application that is based upon the concept of probabilistic reasoning. The application is the first step towards an automated fault management expert system that uses statistical methods to isolate faults in broadband networks. In its core, the inference engine is based on Lauritzen and Spiegelhalter algorithm for computations of probabilities on graphical structure.

In this report we describe the various steps of the knowledge engineering taken toward the composition of a belief network for fault identification on XUNET as well as the design of the inference engine and the data acquisition system. Future research recommendations are outlined at the end.

September 28, 1994.

This document reports the work done at AT&T Bell Labs, Murray Hill, NJ, from May 2 to August 19, 1994, as part of XUNET summer research program.

Probabilistic Reasoning for Fault Management on XUNET

Jean-François Huard
Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University, New York, NY 10027-6699

1. Motivation and project summary

Future telecommunication networks are expected to be giant, fast and complex. As the size and speed of these networks grow, their dynamics become increasingly difficult to understand and control. This will pose significant problems on managing future networks. Fault management will be an especially critical task, since it must ensure the reliability and survivability of the network.

In today's communication networks, the diagnosis of faults is often not too difficult [1]. The knowledge of the network manager combined with the alarms reported is usually enough to localize rapidly most failures. As broadband networks grow in size, fault diagnosis will have to be automated, otherwise these networks will become unmanageable. A single fault can generate alarms from a variety of domain, and many of them are not helpful.

Most of today's fault management systems are based on a deterministic network model. A serious problem is their inability to isolate primary sources of failure from uncoordinated network alarms, which makes automated fault identification a difficult task. We propose to apply probabilistic reasoning, which implies a non-deterministic network model, to allow for automated fault isolation and fault detection in large networks. Probabilistic reasoning can be used as an approach to diagnose network problems in case of uncorrelated, missing, or delayed information. Furthermore, it allows to introduce experience about the operation of the network into the management system, by using the concept of "weighted belief" and "virtual evidence."

In this project, we developed a fault management application that is based upon the concept of probabilistic reasoning. It is the first step toward an automated network fault management expert system that uses statistical methods in its core to isolate faults.

This report is divided as follows. In Section 2 we briefly introduce the fault management task and describe the high level architecture of the system. In Section 3 we introduce the model chosen for the knowledge representation, namely belief networks. In Section 4 we describe the steps taken in the knowledge engineering phase for the composition of a belief network for fault identification on XUNET. The implementation of the inference engine and of the data acquisition system are presented in Section 5. Finally, the conclusion and research recommendations are given in Section 6.

2. Fault management and expert systems

The task of fault management is to detect, isolate, and repair problems in the network and its control systems. The first step, fault detection, is usually implemented within the network protocols and devices. Our work focuses on the second step, which includes fault localization and fault identification. Fault localization is achieved through algorithms that compute a possible set of faults. Fault identification is done by testing the hypothetical faulty component(s). The last step, fault correction is achieved by taking corrective actions. This step may need equipment repair, change of system configuration, or software removal of bugs.

Our approach for identifying faults uses a probabilistic model called belief networks. Belief networks can be described as acyclic influence diagrams, on which a probabilistic model is superposed. When a fault/alarm is detected, a flow is propagated in the belief network, and a set of most probable faults is localized. Once possible faults are known, operations are performed to identify the faulty components or to obtain additional information for further fault localization.

To perform this (iterative) task, we propose an expert system architecture as shown in Figure 1. The core of the system is composed of two parts, an inference engine and a decision engine. The *inference engine* computes some probabilities on the fault model (belief network)

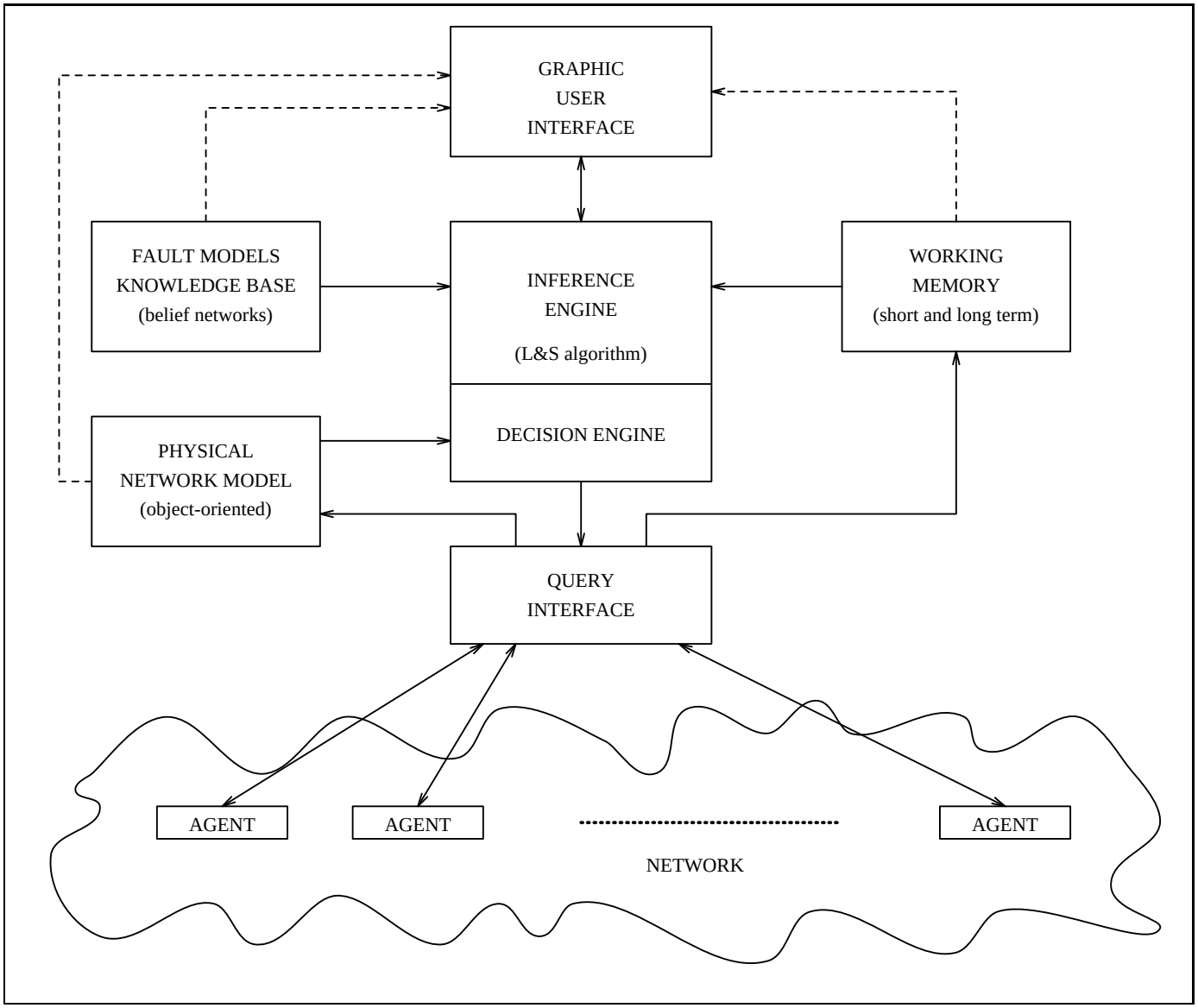

Figure 1: Architecture proposed for the network fault management expert system.

given the accumulated evidence (alarms and responses from queries to the network). It is based on Lauritzen and Spiegelhalter algorithm for computations of probabilities on graphical structure [2]. The inference engine requires some network fault model which is contained in the knowledge base.

The *knowledge base* is an abstraction of the human understanding of network faults propagation and dynamics using causal relationships among various objects of the network. This knowledge is modeled in the form of belief networks.

The *decision engine* is an online algorithm that computes a set of most probable faults and needs to decide which network component should be tested in the next phase of the diagnosis process. Its computation is performed by using the probabilities provided by the inference engine and its decision is based on some cost function where the overall expected cost is to be minimized.

The *network physical model* is a database that abstracts the network resources. It is used by the decision engine to find which network object to query next. It provides a way to bind the "generic" faulty object to the real physical resources to test.

The *query interface* provides the facilities for the management system to acquire evidence about the network state. It is based on the manager/agent paradigm.

The *working memory* accumulates the evidence obtained from the network alarms and from response to queries. It keeps information on the current diagnostic process (short-term basis), but can store information on a long-term basis in order to use learning algorithms for modeling at a later stage.

Finally, the *graphic user interface* presents a visual abstraction of the network faults, faulty components and diagnostic process for a network operator in an informative way.

The system architecture is quite simple, however, several issues need to be looked at closely: the modeling of network fault propagation as belief networks, the binding of physical resources to "generic" faulty network objects, the identification of most probable fault and the online decision algorithm. Part of this system was implemented and its implementation is described in Section 5. First, let us introduce the belief networks modeling and the knowledge engineering that was done on XUNET.

3. Belief networks

In this section, we introduce the model chosen for the knowledge representation, namely belief networks. Ideally, we would like a mathematical model that captures the network as a whole. It should provide the ability to include physical as well as logical objects and provide the basis for various computational studies. Furthermore, in the process of modeling, it should become far more apparent whether some essential variables have been omitted or, whether redundant variables have been inadvertently included. First we define what is a belief network and disscuss why we have chosen this model. Then, we introduce the concept of modeling discrete event systems as belief networks.

3.1 What is a belief network?

For an intuitive introduction to belief (Bayesian, causal) networks the reader is referred to an excellent tutorial paper by Charniak [3]. Belief networks are acyclic influence diagrams on which a probabilistic model is superposed. They model dependency (causal) relationships between different objects or events. The nodes of the diagram are propositional variables and the arcs indicate relationships between the propositions. Each node has associated to it, a set of conditional probabilities that relates it with its neighbors.

The belief network approach has been chosen because we believe that it is a modeling approach that captures compactly the essence of a discrete events system and provides the ability to be refined as the system is better understood. The causal relationships of the system are modeled by assigning propositional variables to each object of interest (physical or logical) and by drawing directed arcs between them to reflect the relationships assumed logical, physical, temporal or conceptual. The arcs also serve to define the underlying probability model [4]. An implicit assumption in these networks is that, given all the direct parents of a node, the node is independent of all the other nodes of the network. This gives rise to an analytically tractable model whose equilibrium probabilities have product form, namely Markov fields. This assumption also reduces the storage requirement of the probabilities. The set of a priori probabilities of the root nodes and the set of conditional probabilities relating neighboring nodes are sufficient to calculate the joint distribution of

the belief network. Formally, we define a belief network as follows [5]:

Definition 1 *Let V be a finite set of propositional variables defined on the same probability space, let $(\Omega, \mathcal{F}, P)$ be their joint probability distribution, and let $G = (V, E)$ be a directed acyclic graph. For each $v \in V$, let $c(v) \subseteq V$ be the set of all parents of v and $d(v) \subseteq V$ be the set of all descendents of v. Furthermore for $v \in V$, let $a(v) \subseteq V$ be $V - (d(v) \cup \{v\})$, that is, the set of propositional variables in V excluding v and v's descendents. Suppose for every subset $W \subseteq a(v)$, W and v are conditionally independent given $c(v)$; that is, if $P(c(v)) > 0$, then*

$$P(v|c(v)) = 0 \quad or \quad P(W|c(v)) = 0 \quad or \quad P(v|W \cup c(v)) = P(v|c(v)).$$

Then $C = (V, E, P)$ is called belief network, and the set $c(v)$ is called the set of the causes (parent) of v.

The next theorem gives us the equilibrium distribution associated with a belief network [5].

Theorem 1 *If $C = (V, E, P)$ is a belief network, then $P(V)$ is given by*

$$P(V) = \prod_{\substack{v \in V \\ P(c(v))>0}} P(v|c(v)),$$

where $\prod_{v \in V, P(c(v))>0}$ means we are taking the product of all propositional variables in V for which $P(c(v)) > 0$.

Other requirements of knowledge representation are easily modeled with belief network. For example, there is no problem in modeling logical links, in which a probability is either 0 or 1, according to the values of the parents. Furthermore, the model is able to support virtual evidences to study the effect of variables taking particular values with a certain degree of uncertainty.

The approach also enables us to track the system as it evolves in time. When the symptom of a fault (or evidence) comes in, the most probable cause of the fault can be deduced—or at least, the set of most probable causes.

3.2 Modeling discrete event systems as belief networks

We illustrate the concept of modeling a discrete event system as a belief network through an example of connection setup taken from [6].

Let us suppose that we have a client/server communication problem where a client tries to set up a connection with a server and cannot access it. The client then reports repeated access-failure to the fault manager. Several causes are potentially responsible for this access-failure: network congestion, link fault or server fault. On the other hand, network congestion is often caused by high traffic load. Also, when a link fault occurs, a link alarm is normally triggered. This situation can be represented by the influence diagram of Figure 2.

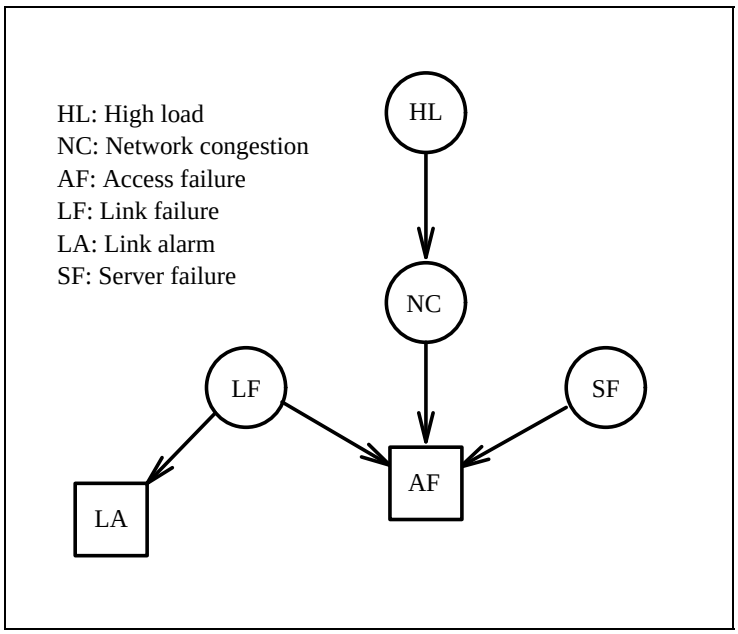

Figure 2: An acyclic influence diagram.

To complete the model, we need to assess certain probabilities. According to the independence assumption mentionned earlier (d-separation principle [7],) the set of a priori probabilities of the root nodes and the set of conditional probabilities relating neighboring nodes are sufficient to calculate the joint distribution of the network using Bayes' law.

The conditional independencies implied by the causal relationships and the restriction that we are not allowed to have cycles in the influence diagram, appear to be the most restrictive assumptions.

An important issue in the modeling, is the probability assessment. In the model, we are assuming perfect knowledge of the conditional probabilities of the influences. In practice, this may not always be the case. However, in case of uncertainty about some evidences, the model can cope with this problem by adding virtual evidence [2, 5].

Other issues must be taken into account. For example, the external and internal constraints that limit the freedom in choosing control, monitoring and optimization criterion. Internal constraints are usually imposed by the system structure itself and arise as a result of restricting the control or measurement process. On the other hand, the external constraints arise from the desire of the manager, not from the physical or structural limitations. External constraints usually involve finite resources. They are imposed from the outside and have nothing to do with the mathematical restrictions induced by the model itself. They are the ones that make the problem physically meaningful and will be incorporated as a "query cost" in the algorithmic problem.

To study the importance of the modeling constraints and assumptions involved in the belief network approach we tried out some network modeling techniques on XUNET.

4. Causal modeling and XUNET belief network

If we illustrate the inference engine (to be described in Section 5) as the heart of the expert system then we could say that the knowledge base is the brain of the system. When the knowledge base is not well designed, then the expert system does not work as expected. In practice, the knowledge engineering is the most difficult part to realize in the development of an expert system because the knowledge engineer is usually not an expert in the field. In particular, for this project, I had to work jointly with XUNET's designers and managers in order to compose the belief network. The steps taken toward the composition of the belief network were as follow:

1. The *identification* of a set of faults that occurs "regularly."

2. The *composition* of individual belief networks for the identified faults.

3. The *construction* of the global belief network for XUNET.

4. The *assignement* of probabilities on the global belief network.

Each step is described in the following sections.

4.1 Identification of a set of faults and their composition

The starting point of the design of our belief network is to identify a set of faults that occur (or has occured) regularly, for which we understand the dynamics. In this way, for the composition of the belief networks, we are able to focus our attention on a single problem at a time. In short, our work becomes more modular and the belief networks become easier to build and to understand. In this project, we identified the following set of faults.

1. *No connection* for the client/server example.
2. *Queue module* in bad mode.
3. *No logfile mail* received at night.
4. *No board seen* in a switch slot.
5. *Configuration* out of date.
6. *Link down.*

The *no connecition* problem is the one described in the example of Section 3.2. On XUNET, this problem happens if the server is dead, if the signalling code that is running is not compatible between some nodes, if too many channel connections are opened, if there is an intermediate hardware problem between the client and the server (because of the XUNET physical topology), and so on.

The *queue module* problem occurs when the receiving channels of a queue module are not in block mode. The default receiving mode of the switch is block mode. However, when a queue module board resets (for some reason), the board is set in cell mode due to the hardware configuration. The block mode has to be set by software. This is done when the control software is downloaded from the switch controller to the queue module successfully. When the software cannot be downloaded properly (or some experimental software is left in

the memory after an experiment), then we say that we have a *queue module* problem. In this case, the packet reassembly does not work properly and large packets do not get thru (however they might, if all cells that compose them arrive contiguously at the buffer). In the case of a bad reassembly, the packet is dropped.

The *no logfile mail* problem has two forms. Every night, the daily logfile of the switch controllers are sent to the management center. In some case, we might not receive all the logfiles. This might be due to an Internet problem, to an XUNET link problem or it might be that the switch controller is not working properly and did not send the logfile. What might also happen, is that we receive the daily logfiles, but the end of day status report is not reported in all of them. This would usually be caused by the absence of some switch monitoring processes that do not dump their report.

The *no board seen* problem happens when a board is not seen (by polling) in a given switch cabinet slot, and is expected to be there. Note however that, unless someone physically removed the board from the switch, we should be able to poll the board. For example, we might not see a specific board when a fuse is blown, when a board does not register properly (done by hardware) or if the insvc (in service) file is corrupted.

The *configuration* problem is common in communication network. It is about the consistency of the network database. In particular, it addresses the network topology update as well as control software version, *etc.* No belief network was composed for this problem. We believe that this specific problem is an entire domain of research. In order to compose a belief network, we need to be more specific about which configuration problem we are facing.

The *link down* problem is the typical problem of hardware failure when some component of a transmission link is out of service. It might be the transmitter or receiver line cards, the physical transmission link itself, *etc.* This problem, like the previous one was not composed. It is not well suited for belief networks modeling because it is too close to the hardware level. In general, problems that fit well in the belief networks approach are the ones with dynamics not very well understood. That is, the ones like *no connection* and *no logfile mail*, where several layers of network protocols are involved.

Once the above problems were undertood, we composed a belief network for each of them. These are shown in Figures 3 to 6.

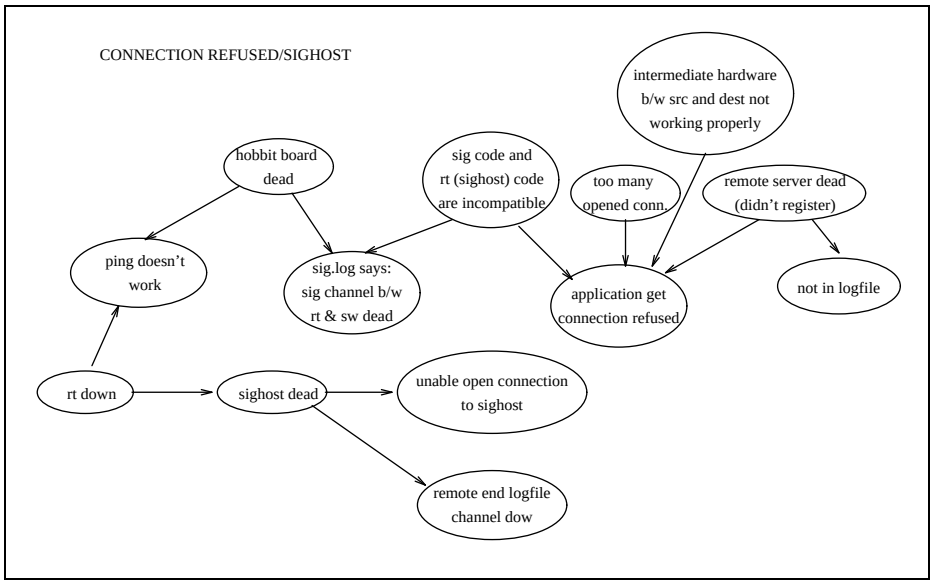

Figure 3: The *no connection* belief network.

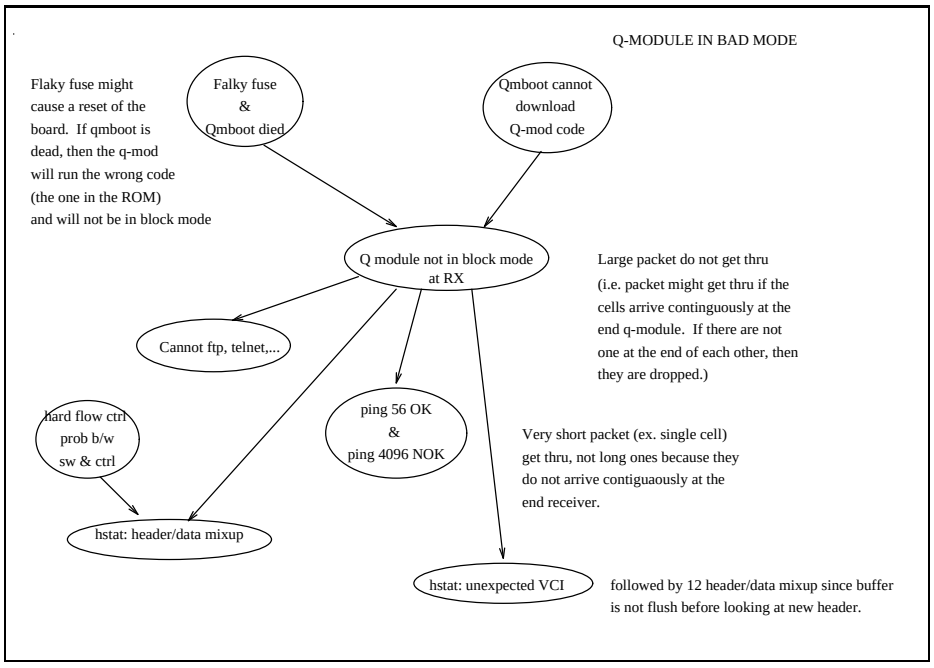

Figure 4: The *queue module* belief network.

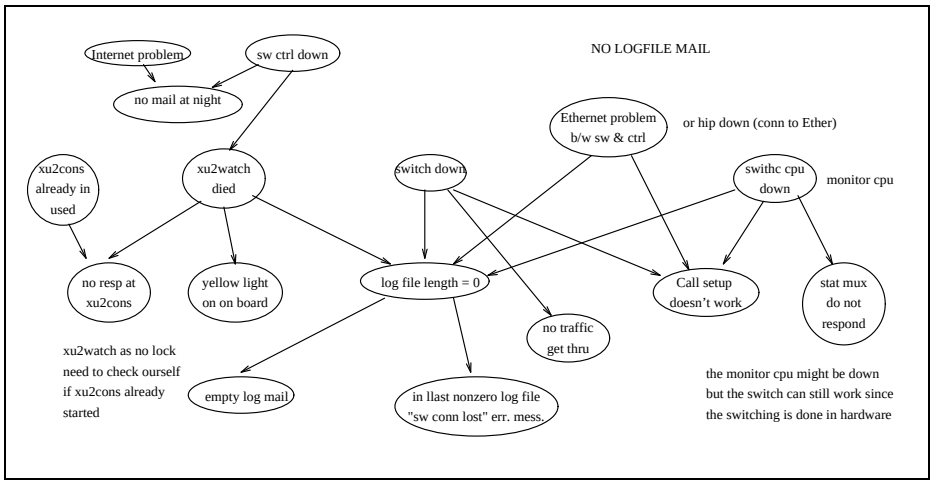

Figure 5: The *no logfile mail* belief network.

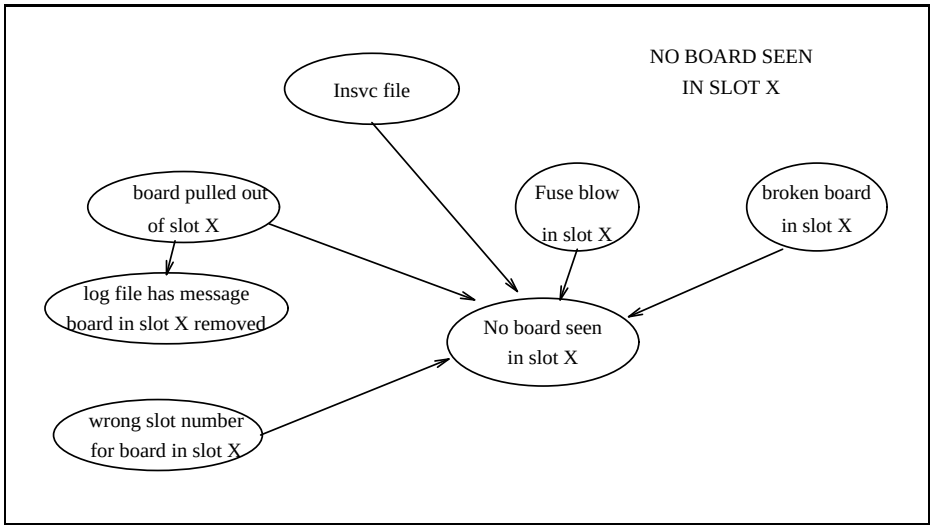

Figure 6: The *no board seen* belief network.

4.2 Construction of the global belief network and probability assessment

The construction of the global belief network was done by taking the union of the individual belief networks of the previous section. In short, the global set of nodes is the union of the individual sets of nodes, and the global set of edges is the union of the individual sets of edges. The resulting diagram was then modified for consistency and complexity consideration by removing, clustering or adding some edges and nodes. Further improvements were done ulteriorly (during the probability assessment phase), again by removing, clustering of adding nodes and edges.

The probability assessment was done on the global belief network, working one problem at a time, and starting by assessing the *a priori* probabilities of all root nodes. Again, this work was done jointly with the system designers and managers. In order to facilitate the probabilities assessment, two ways were used. The first one was to use a scale that assigns a probability interval to quantitative adjectives and adverbs [8] (see Figure 7) and the other one was simply to assign the probabilities directly. When the probabilities were unknown, they were assigned uniformly.

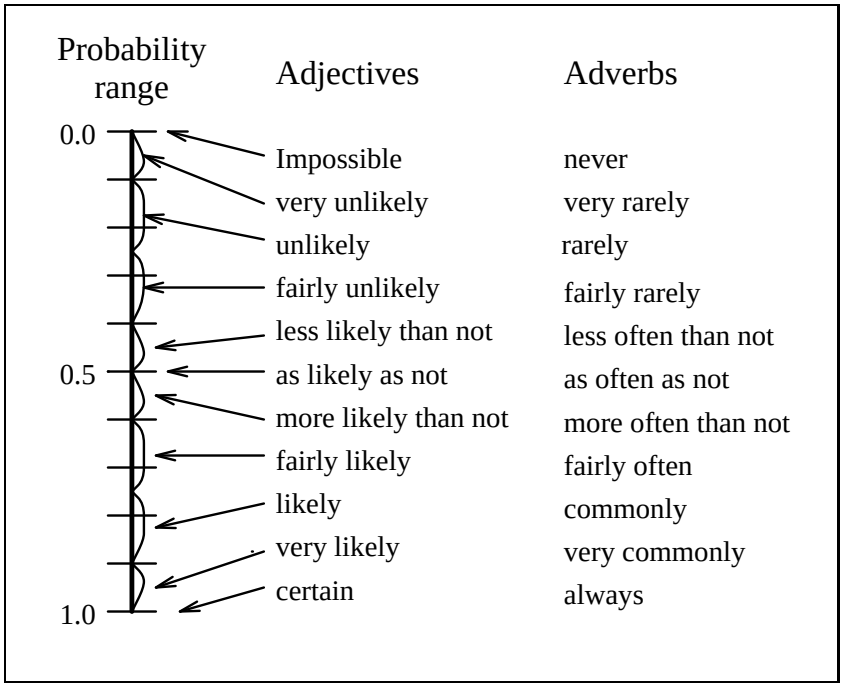

Figure 7: Scale for the assignement of probabilities.

4.3 XUNET belief network

The resulting XUNET belief network obtained from the four steps described in the previous section is shown in Figure 8.

This belief network for fault diagnosis is one of the first attempt (if not the first) to such modeling of a real ATM network testbed. The model obtained still needs to be refined and needs to be validated. The refinement can be done again by adding, clustering, removing nodes and edges, revising the probabilities, *etc.* The addition of new problems to the diagram, would also help to refine the model. For example, when a variable depends on too many parent nodes (say more that 4), then it should be splitted, or it should be defined more precisely. For example, this is what happend with the `callsetup`, `signalling` and `qmboardseen`. Because of the number of parents, the conditional distribution of these nodes become large and difficult to assess. It is worthed mentionning that there exists method and models for the probability assessment that were not used here. For example the noisy-or-gate method when failures are statistically independent. These should be better understood in order to ease the assessment of probabilities.

Furthermore, some consistency tests should be done. For example, if a router is down, then all the processes running on that router are dead with certainty. Therefore, the signalling should not be working. If the probabilities are assessed such that there is some probability that the signalling is working then the model is inconsistent. Moreover, if we confirm that the signalling is working, while it is not supposed to with probability one, then we have a consistency problem, and wrong numerical results. For the above reasons, the model should be validated in some manner. To validate further the probabilities, is would be worth to findout which failures are most likely, or given specific symptoms, which are the most likely failures and compare these results with the managers experience.

Finally, the method learned (and developed) during the design of this belief network is similar to the probabilistic similarity networks[1] approach [9]. It is proposed to try out this method of modeling over an extended set of problems.

[1]This approach was unknown to the author at the beginning of the project and was found out during the course of the work.

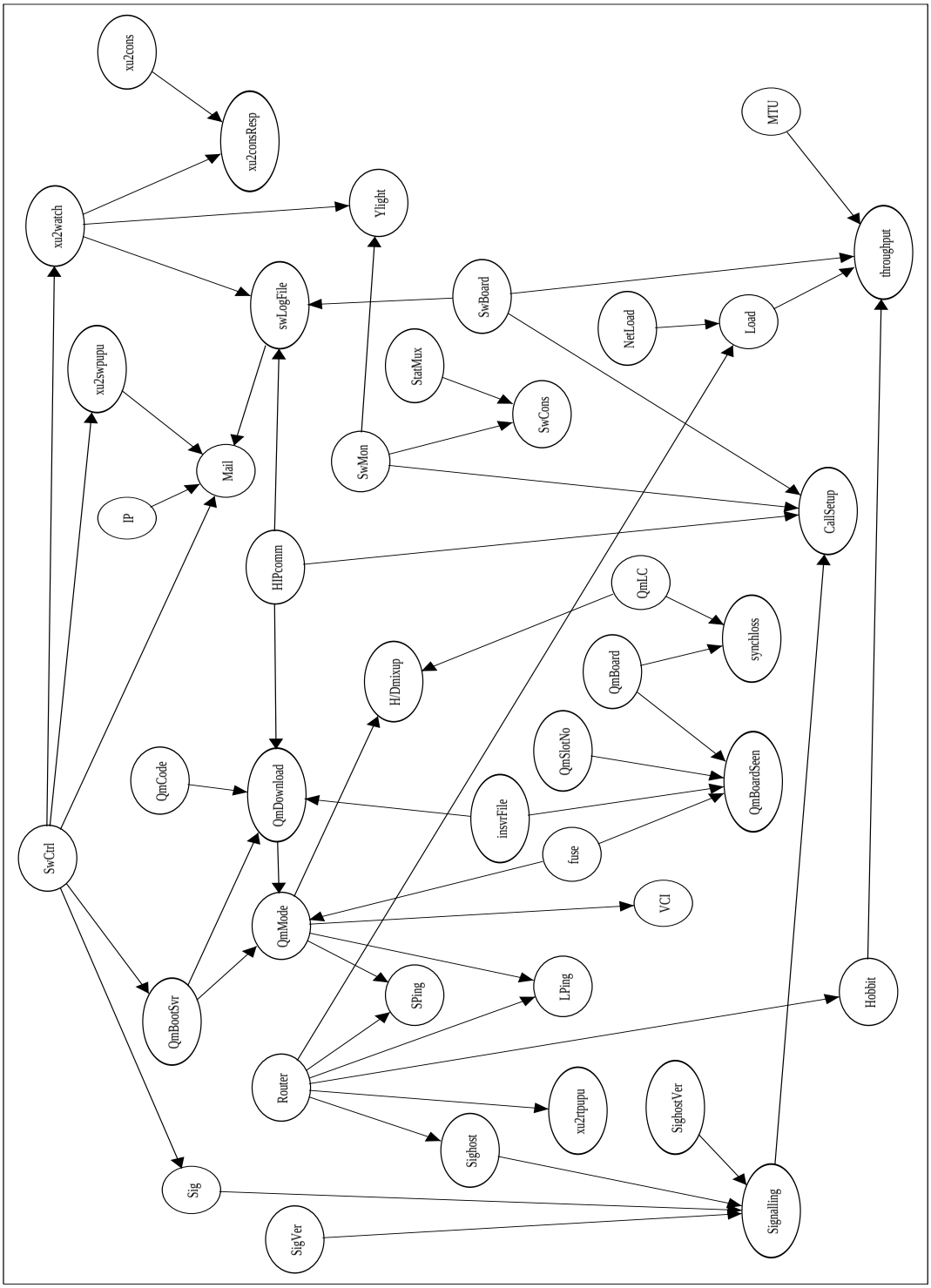

Figure 8: The constructed global belief network for XUNET.

5. Implementation

In this section we describe the parts of the expert system architecture proposed in Section 2 and depicted in Figure 1 that were implemented.

First, an editing tool to enter the belief network was designed. Then a preprocessing program to derive a good machine representation for the computation of probabilities in the belief network was implemented. This new representation is called a tree of cliques. An inference engine (with a graphic user interface) was then implemented to perform the probability computations on the tree of cliques. Finally, the query interface was emulated by writing Unix shell scripts for each of the variables in the belief network.

5.1 CADET and id2cn

CADET is an acronynm for Computer Aided Decision Evaluation Tool. It is an AT&T software that uses influence diagram to help decision maker in evaluating risk in their decision. Belief networks are a special type of influence diagrams with only chance nodes. Basically, what CADET provided us with, is a graphic interface to enter the belief network data on file. To this mean, we modified the output routines to provide us with the required information, and so that CADET files remain compatible with later versions. We then implemented a filter (`id2cn`) to parse CADET output files (`.id`) to a convenient file format (`.cn`) for our preprocessor.

5.2 Generating a tree of clique from a belief network—cn2tc

The machine representation that our inference engine uses for the computation of probabilities is based on Lauritzen and Spiegelhalter algorithm [2]. The reader is referred to the original paper or to [5] for a complete description of the transformation process. In short, we start with a belief (causal) network, and perform some operations to keep consistency in the probability distribution. First we marry the parent of each node, that is, for each node, we connect all of its the parents together. Then we remove orientation on the edges, and we call this graph the *moral* graph. From that point, we are guaranteed that our probability distribution is consistent; whatever edge we might add will not affect the probability

distribution. The next step is to find all the cliques of the moral graph. However, this is a hard problem. To facilitate this search, we triangulate[2] the graph [10]. Triangulating a graph can be done in linear time (in the number of edges and nodes). Then, we can find the clique in linear-time since clique determination on triangulated graphs is linear (also in the number of edges). Once we have a set of cliques, we create a tree of clique and its potential representation from the original probability assessment.

This program is simply a "filter." It reads a belief (causal) network file (`.cn`) and generates a tree of clique representation (`.tc`). The idea was that, even though it might require a lot of time to generate the tree of clique, it would not matter since, this mathematical structure is more efficient for the probability computations. The initialization is only done one time once the belief network is constructed, so we were willing to pay the price in order to perform better in the real-time diagnosis task at the later stage. It was written separately from the inference engine, since some of the steps involved in the computation of the tree of clique can be quite computationally demanding (several steps are NP-hard problems). Nonetheless, the program runs surprisingly fast. On XUNET belief network (42 nodes and 53 edges), it moralizes, triangulates, finds the 24 cliques, creates the tree of cliques and computes the potential representation in just the time of hitting the return key.

5.3 Inference engine—tc and Xtc

As mentionned earlier, the inference engine is based on Lauritzen and Spiegelhalter algorithm [2]. It was implemented with no restriction except that the probability distribution of needs to be discrete.

The engine has a graphical user interface to ease its usage. Currently, the functionalities are minimal. We can instantiate nodes (assess with certainty the actual state of a node), propagate the evidence (compute the new probability distribution) and display the marginal ditribution of each node. For example, we can instantiate the nodes *router* and *swctl* to *running*, and get the probability that the signalling is working given that the routers and switch controllers are working.

[2] Triangulated graphs are also known as chordal graphs.

Furthermore, the software is written so that we can accumulate evidence before propagating it; that is, we do not need to recompute the probability distribution every time that new facts arrive.

In the near future, the interface and the functionalities will be enhanced. Not only the marginal probabilities will be available, but also any combination of joint distribution. Virtual evidence (that represent our trust in the truth of evidence) will also be implemented. Finally, fancy interface features such as font and color selection, printing, editing, *etc.* will be implemented.

5.4 Query interface and binding physical resources—UNIX shell scripts

For reason of shortage of time, the query interface and the binding of the physical resources to the belief network was emulated by writing a Unix shell script for each variable. For example, if we would like to know the status of the switch controller at the central office of Chicago, it suffices to enter the command `Swctl chcg` at the prompt of any XUNET machine. A brief status report such as *up* or *down* would then be outputed. Whenever possible, the shell scripts use the XUNET control channels to perform their queries; otherwise, they might use the Internet. Therefore, it assumes that there is a way to communicate to each node of the network. This seems a reasonable assumption since the management center should have some way to talk to each of agents. When it cannot use both of the control channels and the Internet, then the statmux would need to be used. However, this functionality was not automated.

Furthermore, the binding of the physical resources to the belief network is done by the human manager when the shell script command is typed at the keyboard. The sole problem of binding physical resources to the belief network is an entire research domain. It was therefore thought that having an interactive query interface would provide some functionality of the entire system faster. Finally, shell scripts were easily written in a short time period, and will be easy to integrate at a later stage using the Unix C library function, `system(3)`.

6. Conclusion

In this project, we developed a fault management application that is based upon the concept of probabilistic reasoning. A proposal for a network fault management expert system architecture was depicted in Section 2. Some parts of the system were implemented during the course of this project and were briefly described in the previous sections. In particular, the inference engine was implemented following the Lauritzen and Spiegelhalter algorithm for the computation of probabilities on graphical structure.

Several components of the architecture need to be studied further. In particular, we need to study how to bind the physical network resources to the belief network modeling. Should (OSI) agents provide the required information upon request? Should we have a distributed or central database? How much intelligence should be given to the agents?

Another issue is the decision engine algorithmic problem; that is, to find an online algorithm that computes a set of most probable faults, and needs to decide which network component should be tested in the next phase of the diagnostic process based on some cost function. Furthermore, what is the most likely diagnosis? [11].

The last issue concerns the XUNET belief network that needs to be refined and validated. For the refinement of the model, the probabilistic similarity networks approach seems interesting. It provides us with a methodology to construct a belief network by composing small belief networks. It is very similar to the approach that was taken in this project to construct the XUNET belief network. Finally, no matter which method is chosen for the modeling of discrete event systems as belief networks, a methodology to validate the belief network needs to be found.

Acknowledgements

I would like to thank all the XUNET team. In particular, Chuck Kalmanek, Pat Parseghian and Bill Marshall for their help during the modeling phase. Gregg Vesonder for stimulating discussions, Kazuo Ezawa for giving me access to CADET and Dan Dvorak for getting me started with the modeling at the early stage of the project. Finally, thanks to all the members of the center 1127 for the stimulating working environment.

References

[1] N. Dawes, J. Altoft, and B. Pagurek, "Network diagnosis by reasoning in uncertain nested evidence spaces," *IEEE Trans. Commun.*, vol. 43, no. 2–4, pp. 466–476, 1995.

[2] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *J. R. Statist. Soc. B*, vol. 50, no. 2, pp. 157–224, 1988.

[3] E. Charniak, "Bayesian networks without tears," *AI Magazine*, pp. 50–63, Winter 1991.

[4] T. P. Speed, *Influence Diagrams, Belief Nets and Decision Analysis*, ch. 3, pp. 49–63. New York: John Wiley & Sons, 1990.

[5] R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* New York: John Wiley & Sons, 1990.

[6] A. A. Lazar, W. Wang, and R. H. Deng, "Models and algorithms for network fault detection and identification: A review," in *Proc. IEEE Int'l Conf. Commun.*, (Singapore), pp. 999–1003, November 16–20 1992.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, California: Morgan Kaufmann Publishers, 1988.

[8] M. Henrion and M. J. Druzdzel, "Qualitative propagation and scenario-based schemes for explaning probabilistic reasoning," in *Uncertainty in Artificial Intelligence 6*, (Amsterdam, The Netherlands), pp. 17–32, Elsevier Science Publishers B.V., 1991.

[9] D. Heckerman, *Probabilistic Similarity Networks.* Cambridge, Mass.: MIT Press, 1991.

[10] J.-F. Huard, "Chordal graphs: Their testing and their role," May 1994.

[11] D. Poole and G. M. Provan, "What is the most likely diagnosis?," in *Uncertainty in Artificial Intelligence 6*, (Amsterdam, The Netherlands), pp. 89–105, Elsevier Science Publishers B.V., 1991.