A domain independent data structure for telecommunications using adapted ATMS

Weiru Liu
School of Info. and Soft. Eng., Univ.of Ulster at Jordanstown,
Co.Antrim, BT37 0QB, N.Ireland, UK.
e-mail:w.liu@ulst.ac.uk

May 12, 2003

Abstract

In this paper, we investigate into applying a novel approach to representing the topologies of telecommunication networks (TNs) by adopting the ideas in the assumption-based truth maintenance system (ATMS). A new meaning is given to the justifications to reflect the requirements of telecommunications. As results, a domain independent data structure is designed for telecommunication networks using which the maintenance of the topology of a constantly changing network needs only little work, and a generic fault detection algorithm is developed.

Keywords: ATMS, diagnostic systems, telecommunications.

1 Introduction

The applications of telecommunications are increasingly popular in recent years. The telecommunication networks (TNs) in use are getting larger and larger. Therefore, the management of such a network is increasingly difficult. Fault management, a crucial functionality in any network management, is even more difficult to handle. Inevitably, various kinds of approaches for fault management have been investigated, such as model-based approaches ([4], [6]), neural nets methods (e.g., [5]), fuzzy-set based approaches (e.g., [7]), and probability theory ([1]) or evidential theory related approaches ([2]).

When one uses a model-based diagnostic technique, two main components are necessary. One is the structural model which simulates the physical model of the network, another is the behavior model which simulates how the network works. This method has been used widely in the network maintenance in the past. However, researchers gradually realized that this method has a number of disadvantages, such as, the difficulty of redefining the structural model when the network has changed, the difficulty of acquiring network behavior knowledge from experts, and the exponentially computational complexity with the number of components. Some model-based approaches even require an exhaustive model of the fault behavior which turns out to be difficult to define. Therefore, searching for alternative techniques is inevitable.

The neural network approach can be seen as an alternative to the model-based approach in the way that each fault detection only involves a number of components in the network. However, as the procedure of building (training) a neural network is very slow, and the interaction of such a system with human experts is difficult to master, the method has not been widely used in this area.

Dawes et al ([2]) discussed the method of using the Dempster-Shafer theory of evidence to deal with the uncertain status of components (broken or not) and to manage the degrees of beliefs about their status according to the fault reports. In this paper, a list of nodes describing all components (or called devices) is defined. In particular, within each node, there is a slot recording the distance (the count of how many nodes are in the middle) between this node and the nearest data collector (monitor). This information is combined with the knowledge of network topology to decide to which nodes the propagation proceeds. There are two limitations in their method. First, if a network is changed (updated) in some way, many nodes would have to be revised to reflect the changes. Second, the knowledge about the network topology has to be changed. These two tasks may involve considerable time and effort along with any changes of the network.

In order to overcome these difficulties, in this paper, we introduce a novel approach to designing a domain independent data structure for telecommunications using assumption-based truth maintenance systems (ATMSs) [3]. An adapted ATMS data structure is proposed to describe the topology of a TN. In this data structure, a list of nodes are designed for devices in a network which are similar to the nodes in [2]. However, our method is superior to [2] in the way that we don't record the distance of a node to a monitor, rather we record the neighbour nodes of a node. So that the topology of a network is implied in all these nodes, and no separate knowledge about the structure of a topology is required. Therefore, the changes needed to reflect the revised networks are minimum.

The main contributions of this paper are:

1) Providing a simple, but efficient method to represent the topology of a network.

2) Extending or changing the network structure only need to revise the messages on connections in a limited number of nodes.

3) Enabling a system using this data structure to deal with uncertain and imprecise fault messages by associating probabilities on proper nodes.

4) Designing a domain independent fault detection algorithm for locating possible sources of faults.

Besides, the data structure also support the parallel execution of the algorithm in the case of multiple faults.

Therefore, the new approach has a number of advantages over the methods in [4], [5], [2]. It has promising application potentials in the telecommunication network management domain.

The paper is organized as follows. Section 2 introduces the basics of an ATMS. Some key concepts *justifications*, *assumptions*, and *labels* are explained in detail. Section 3 discusses how to adapt the ATMS structure to describe the topology of a network. New meanings are given to the justifications to suit the features in networks. A special kind of fault problem, *i.e.*, connection failures, is used to demonstrate how to use the data structure. Section 4 introduces the generic fault detection algorithm based on this data structure and Section 5 summarizes the paper.

2 The Basics of ATMSs

The assumption-based truth maintenance system (ATMS) [3] is a symbolic approach to manipulating statements in order to obtain a set of specific statements in which we believe. The basic and central idea in such a system is that each statement is assigned to a set of reasoning pathes, each of which is called a *justification*. Each justification contains a number of other statements from which the current statement can be derived. Justifications are specified by the system designer. Through justifications, a set of supporting environments is produced for the statement. A supporting environment contains a set of special statements which are assumed to be true if no conflicts, and are called *assumptions*. Each supporting environment suggests that the statement is believed to be true under these assumptions. In an ATMS, each statement is represented using a unique *node*.

For instance, if we have two statements representing inference rules $r_1 : p \to q$ and $r_2 : q \to r$, then logically we can infer that $r_3 : p \to r$. In an ATMS, if r_1, r_2 and r_3 are represented by $node_1$, $node_2$ and $node_3$ respectively, then $node_3$ is derivable from the conjunction of $node_1$ and $node_2$. We call $(node_1, node_2)$ a justification of $node_3$. So a justification is a set of nodes from which a node is provable. If we also have $node_4$ and $node_5$ from $r_4 : p \to s$ and $r_5 : s \to r$ respectively, then $(node_4, node_5)$ is another justification of $node_3$. Normally a statement may have several justifications.

If we take p and r as two devices in a network, then formula $p \to r$ can be explained as that p and r are connected if pairs p and q, and q and r are connected (or alternatively, if pairs p and s, and s and r are connected). Furthermore if $node_1$ and $node_2$ are valid under the conditions that A and B are true respectively, then $node_3$ is valid at least when the condition $A \wedge B$ is true, denoted as $\{A, B\}$. $\{A\}, \{B\}$ and $\{A, B\}$ are sets of supporting environments of $node_1, node_2$ and $node_3$ respectively. Associating $node_3$ with the supporting environments (such as $\{A, B\}$) and the justifications (such as $(node_1, node_2)$), $node_3$ is represented in the form of

$$< node_3 : p \to r, \{\{A, B\}, \{C, D\}\},$$

$$\{(node_1, node_2), (node_4, node_5)\} >$$

where $\{C, D\}$ represents the set of supporting statements derived from justification $(node_4, node_5)$. $node_3$ is true when A and B are true, or when C, D are true. The collection of all possible sets of supporting environments is called the *label* of a node, and denoted as $L(node_i)$.

In general, a node in an ATMS is in the form of $< node_i : datum, label, justifications >$. If we assume that r_1, r_2, r_4, and r_5 hold without requiring any dependent relations on other nodes, then $node_1$, $node_2$, $node_4$ and $node_5$ are represented as

$< node_1 : p \to q, \{\{A\}\}, \{()\} >$,
$< node_2 : q \to r, \{\{B\}\}, \{()\} >$,
$< node_4 : p \to s, \{\{C\}\}, \{()\} >$,
$< node_5 : s \to r, \{\{D\}\}, \{()\} >$.

A, B, C, and D are assumptions which are assumed to be true if there is no conflict. Assumptions are in capital letters in an ATMS and we follow this convention in this paper.

Statement A supporting $p \to q$ can be explained as that devices p and q are connected through cable A. Only when A is working properly, the part of the network is OK. Similarly, B, C, and D can all be explained as cables (or wires). Using the first justification of $node_3$, supporting environment $\{A, B\}$ is obtained. Similarly, the second part of $L(node_3)$ is derivable. Therefore, we can infer the label for any node as long as its justifications are known. A node is believed if at least one of its environments containing assumptions is derived through a justification.

3 The Adapted ATMS Structure for TNs

The main idea in an ATMS is to use justifications as inference pathes to infer the potential conclusions, given the initial inputs. If we apply this idea directly to the fault detection, we will have to enumerate different combinations of phenomena and faults. Obviously, in a large network, this is almost impossible. So we will have to think to apply the idea in ATMSs in a different way. This section concentrates on exploring the new usage of justifications in establishing an adapted ATMS structure for representing the topology of a network.

3.1 New meanings of justifications

Example 1: Assume we have three PCs connected as shown in Figure 1a. If PC_3 is in control of this part of the network (PC_3 is a monitor) and we use each node for each device, the three ATMS nodes representing these three PCs will be as shown in Figure. 1b.

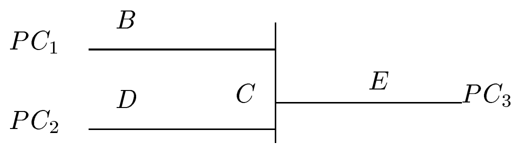

Figure 1a. Three connected PCs.

$< node_1 : PC_1, \{\{PC_1, B, C, E\}\}, \{()\} >,$
$< node_2 : PC_2, \{\{PC_2, D, C, E\}\}, \{()\} >,$
$< node_2 : PC_3, \{\{PC_3\}\}, \{()\} > .$

Figure 1b. ATMS nodes.

Here B, D, and E are cables, and C is a part of a network. $\{PC_1, B, C, E\}$ represents that PC_1 is connectable when PC_1 and all B, C, E are working properly.

If the data collector ([2]) or a alarm residented in PC_3 reports losting contact with PC_1, it implies that the possible fault lies in $\{B, C, E, PC_1\}$. In this case it is obvious to a human expert that checking whether PC_2 is working will help to locate the sources of a fault. That is to see whether the fault is caused by a common part shared by the routes between PC_1 and PC_3 and between PC_2 and PC_3. This will help to narrow down the sources of the fault to either $\{PC_1, B\}$ or to $\{C, E\}$. How can we let a system know how to narrow down the possible sources of faults in general? If we put $node_1$ in the justification part of $node_2$ and put $node_2$ in that of $node_1$ as below

$< node_1 : PC_1, \{\{PC_1, B, C, D\}\}, \{(node_2)\} >,$
$< node_2 : PC_2, \{\{PC_2, E, C, D\}\}, \{(node_1)\} >,$

and ask the system to test the nodes in the justification set. Then this will let the system know which node should be tested on. Surely in this case, a justification is not used to derive a label. Rather, it is used to narrow down the sources of faults among assumptions in a label, because the label sets of $node_1$ and $node_2$ are overlapped. The test on the nodes included in such a justification helps to decide whether the fault lies in the overlapped parts of the environments.

Example 2: We expand the above network by attaching PC_4 with PC_1 through a cable F as shown in Figure 2 (in Appendix). When PC_4 is not contactable, checking PC_1 would be vary helpful for finding the sources of a problem. So a meaningful ATMS node for PC_4 should look like:

$< node_4 : PC_4, \{\{PC_4, F, L(node_1)\}\}, \{(node_1)\} > .$

The justification $(node_1)$ indicates that $node_1$ is in control of linking PC_4 and PC_3. If the connection between PC_4 and PC_3 is down, by looking at the justification of $node_4$, the system knows that checking the connection between $node_1$ and $node_3$ would help to decide the source of a fault.

The function of justification $(node_1)$ in $node_4$ is also to narrow down the source of a fault. However the functionalities of $node_1$ as a justification in $node_4$ and in $node_2$ are different. To distinguish these two types of justifications, we divide the nodes in justifications of $node_i$ into two groups: the essential group and the assistant group.

The essential group of $node_i$: this group contains those nodes, which provide different communication pathes between $node_i$ and the data collector (or central control). The labels of these nodes are parts of the environments of $node_i$. $node_1$ is an essential node of $node_4$.

The assistant group of $node_i$: this group contains those nodes, which share a part of the communication link with $node_i$, but the connection between $node_i$ does not go through any of them. The labels of these nodes are overlapped with some environments of $node_i$, but not subsumed by the environments of $node_i$. $node_1$ is an assistant node of $node_2$.

In this way, the justifications are not used to infer labels as in traditional ATMSs, rather they are given fresh meanings as providing nodes for auxiliary tests in order to decide those assumptions which are likely to cause the fault.

3.2 The adapted ATMS data structure

Upon the new meanings and functionalities of justifications, the basic data structure, therefore, for a TN is an adapted ATMS node, which has the similar structure with a normal ATMS node but has two separate justification sets.

Definition 1 *: Adapted ATMS data structure.*
An adapted ATMS data structure is a quintuple

$$< node_i : device\_name, label, J_1, J_2 >$$

where

- *$node_i$ is a unique identifier to identify a node in an ATMS system,*
- *device_name is the actual name of that device,*
- *label is a set of environments, each of which provides a link between this device and the monitor,*
- *J_1 means* justification 1 *which is a set containing all the essential nodes of $node_i$,*

- J_2 *means* justification 2 *which is a set containing all the assistant nodes of* $node_i$.

When $node_i$ is out of contact, the system choses the nodes in $justification_1$ to test on first, then in $justification_2$.

Every device in a network is mapped to a node in the above data structure, so the connections among all the devices are implied in these nodes. In Figure 2, image that PC_1 is linked with PC_3 via another device D_j, D_j certainly is essential to the link between PC_4 and PC_3. However, we don't take D_j as an essential node for PC_4 in order to keep the sizes of justifications minimum. Doing so will not lose any necessary information for PC_4 because $L(PC_1)$ is a part of an environment of PC_4 and in turn $L(D_j)$ is a part of an environment of PC_1, so $L(D_j)$ is a part of $L(PC_4)$. The following definition decides all the essential nodes for a device.

Definition 2 *: Essential nodes of a node.*

All the essential nodes for node D are included in set $justification_1$ which is defined as:

$justification_1 = \{D_j \mid L(D_j) \subset L(D),\ \forall\ \ L(D_i) \subset L(D) \wedge i \neq j,\ L(D_j) \not\subseteq L(D_i)\}$.

Here $L(D_i) \subset L(D)$ means that for each environment $E(D_i)$, there exists an environment $E(D)$ where $E(D_i) \subset E(D)$.

So the essential nodes of node D are those nodes which form a part of the links between D and the monitor, and are closest to D. Assistant nodes normally only exit for those nodes which are directly connected with the monitor without passing through any other nodes. If $node_i$ is a node staying far away from a monitor, then there must be some nodes sitting in the middle between $node_i$ and the monitor, so $node_i$ has some essential nodes providing vital links.

Fault reports cannot always be precise. In this data structure, if we attach numerical degrees of belief to assumptions, we may be able to infer a statement with a degree of belief. For example, if we know that cables F, B and E all have probability 0.2 to be down independently and the probability of PC_4 is not working is 0.1, then the probability of PC_4 is connected is 0.46 [8]. Therefore, deal with imprecise and uncertain fault reports are possible in this structure.

3.3 An example

To apply the adapted ATMS mechanism to fault management in TNs, the first step is to construct a list of nodes which describe the topological structure of the TN. Within this network, we assume that there is a central control which receives reports from monitors (or called alarms in some cases) across the network. A report from a monitor may either indicate a normal situation or a fault(s).

Example 3: We apply our approach to an example in [2] to show how to use this method in practice. Assume that a part of a network is shown as in Figure 3[1] in Appendix.

Letters $A, B, C, D, E, F, G, H, I, J, L$ and R are links (maybe wares, or cables) between two devices and they are known as assumptions. M_i means that this is the ith monitor of the network. R_1, R_2, R_3, and R_4 are routers. L_2 and L_4 are LANs (we treat each of them as a whole unit). $W_1, ...W_5$ are Workstations.

Using the data structure we designed, a list of adapted ATMS nodes are obtained as follows.

$< node_1 : M_i, \{\{M_i\}\}, \{()\}, \{()\} >,$
$< node_2 : R_1, \{\{R_1, A\}\}, \{()\}, \{()\} >,$
$< node_3 : R_2, \{\{R_2, B, L(node_2)\}\}, \{(node_2)\}, \{()\} >,$
$< node_4 : R_3, \{\{R_3, C, L(node_2)\}\}, \{(node_2)\}, \{()\} >,$
$< node_5 : R_4, \{\{R_4, F, L(node_3)\},$
$\{R_4, E, L(node_4)\}\}, \{(node_3), (node_4)\}, \{()\} >,$
$< node_6 : L_2, \{\{L_2, D, L(node_3)\}\}, \{(node_3)\}, \{()\} >,$
$< node_7 : L_4, \{\{L_4, J, L(node_5)\}\}, \{(node_5)\}, \{()\} >,$
$< node_8 : W_1, \{\{W_1, G, L(node_6)\}\}, \{(node_6)\}, \{()\} >,$
$< node_9 : W_2, \{\{W_2, H, L(node_6)\}\}, \{(node_6)\}, \{()\} >,$
$< node_{10} : W_3, \{\{W_3, I, L(node_6)\}\}, \{(node_6)\}, \{()\} >,$
$< node_{11} : W_4, \{\{W_4, R, L(node_7)\}\}, \{(node_7)\}, \{()\} >,$
$< node_{12} : W_5, \{\{W_5, L, L(node_7)\}\}, \{(node_7)\}, \{()\} > .$

When monitor M_i reports that R_4 is not responding, the high-level control system will load this list of nodes into the system first and then apply an appropriate fault detection algorithm (see next section) to find the sources of the fault. Assume that the serial lines E and F are down at the moment which have caused the links blocked. The algorithm first searches the nodes in J_1 of R_4. There are two nodes $node_3$ and $node_4$ in that set. The algorithm then tests whether $node_3$ is responding OK. The result is yes. So assumption E is taken as a possible source of fault, and put into $F(R_4)$. Similarly, the test on $node_4$ is positive, so F is added as a possible source of the fault as well, and put into $F(R_4)$. There are no more nodes left to be tested. The algorithm outputs E and F as the possible sources of the fault.

4 Fault detection algorithm

Based on the adapted ATMS data structure in Section 3, a generic fault detection algorithm is built. Given a list of nodes which are divided into sub-lists, we assume that each sub-list is led by a monitor node for this sub-network.

[1]We made some changes to make all the devices and assumptions more obvious.

When the central control system receives an abnormal message from monitor M reporting the connection failure of device A (or A is not responding within a certain threshold θ), the following fault detection algorithm is called to isolate the fault sources. The following variables will be used throughout this algorithm.

- $J1$ is a list containing the essential nodes in $justification_1$.
- $J2$ is a list containing the assistant nodes in $justification_2$.
- $F(A)$ is empty initially. This set is gradually expended to contain only those assumptions which may have caused the failure.
- $E(A)$ is used to stand for an environment in the label of A.
- If $E(A) = \{P, ...R, L(J_i)\}$, then $E'(A) = E(A) \setminus L(J_i)$.
- $F'(A)$ is a temporary variable to record those assumptions which may have caused the failure .

We also assume that procedure $Test(M, J_i)$ is a standard program to test whether node J_i is responding to monitor M. If yes, the output of the test is *true*, otherwise, the output is *false*.

Algorithm $FD(M, A, F(A))$: Fault Detection algorithm

Step 1: Load the sub-list of nodes led by monitor M.

Step 2: Search the list to find the node which stands for device A.

Let $J1 = justification_1$ and $J2 = justification_2$ of A.

Let $F(A) = \{\}$ and $F'(A) = \{\}$.

If $J1 \neq \{\}$, go to Step 3; else if $J2 \neq \{\}$, go to Step 6.

Define $F(A) = L(A)$. Go to Step 9.

Step 3: Chose a justification J_i from $J1$.

Let

$$E'(A) = E(A) \setminus L(J_i) \mid L(J_i) \in E(A),$$

$$J1 = J1 \setminus \{J_i\}.$$

Call $Test(M, J_i)$.

Step 4: If $Test(M, J_i) = false$, then call Algorithm $FD(M, J_i, F(J_i))$ and let $F(A) = F(A) \cup F(J_i)$ else define $F(A) = F(A) \cup E'(A)$.

When $J1 \neq \{\}$, go to Step 3.

Step 5: Output list $F(A)$ (each element of $F(A)$ causes one route blocked). If $J_2 = \{\}$, terminate the algorithm.

Step 6: Chose a justification J_i from $J2$. Call $Test(M, J_i)$.

$$J2 = J2 \setminus \{J_i\}.$$

If $Test(M, J_i) = false$, then call Algorithm $FD(M, J_i, F(J_i))$ and let $F'(A) = F'(A) \cup F(J_i)$ else go to Step 7.

Go to Step 6 if $J2 \neq \{\}$ else go to Step 8.

```
Step 7: For every $E(A)$ do
          for every $E(J_i)$
            do {let $E'(A) = E(A) \cap E(J_i) \mid E(J_i) \cap E(A) \neq \{\}$,
                let $F'(A) = F'(A) \cup E'(A)$. }
        When $J2 \neq \{\}$, go to Step 6.
```

```
Step 8: For every two element $f_1$ and $f_2$ of $F'(A)$ do:
          if $f_1 \subseteq f_2$, let $F'(A) = F'(A) \setminus f_1$;
          if $f_2 \subset f_1$, let $F'(A) = F'(A) \setminus f_2$.
```

Step 9: Output list $F'(A)$. Terminate the process.

The algorithm has been implemented both sequentially and parallelly (in terms of runing multiple copies of the algorithm), and a set of testing results is reported in [9].

5 Conclusions

This paper presents some preliminary research results on applying ATMS structure into fault management in telecommunication networks. The paper's main contributions are the adapted ATMS data structure for devices and the generic fault detection algorithm. In this way, the topology of a network is easily built with the data. It is expected that this structure will provide some fresh ideas in combining artificial intelligence methods into traditional fault diagnosis. We narrowed our attention to problems of links between hardwares only in this paper. Principlely a node can stand for a variety of things, such as, a piece of software, a hardware, a LAN as a whole, a cable, a switch, etc. We will investigate into using nodes for softwares in the near future.

References

[1] Blom, M. and Lippolt, B., Diagnosing intermittent faults in telecommunication networks. *Proceedings of Globecom92 Communication Global Uses.* Vol1-3, Ch349, PP544-548, 1992.

[2] Dawes, N., J.Altoft, B.Pagurek, Network Diagnosis by Reasoning in Uncertain Nested Evidence Spaces. *IEEE Trans. on Communications*, Vol.43, No2/3/4:466-476. 1995.

[3] de Kleer,J., An assumption-based TMS. *Artificial Intelligence* 28 (1986) 127-162.

[4] Kehl, W. and H. Hopfmuler, Application of Model-based Reasoning to the Maintenance of Telecommunication Networks. *Proceedings of Industrial Engineering Appl. Art. Int. Exp. S.*:79-88, 1992

[5] Koivo, H, N., Artificial Neural Networks in Fault Diagnosis and Control, *Control Engineering, Practice.* Vol 2, No1:89-101. 1994.

[6] Frank, P. M., Principles of Model-Based Fault Detection. *Proceedings of IFAC Artificial Intelligence in Real-Time Control*:213-220, 1992.

[7] Lee, K. W. and H. N. Lan, Fault Detection and identification Using Neural network and Fuzzy Logic, *Proc. of IFAC Emerging Intelligent Control*:185-190. Hong Kong. 1994.

[8] Liu, W. and A.Bundy, Constructing probabilistic ATMS Using Extended Incidence Calculus. *I. J. of Approximate Reasoning*: Vol 15: 145-182. 1996.

[9] Nigel Wells, W. Liu and K Adamson, Using the ATMS for fault management in telecommunication networks. *in this proceedings*, 1998.

Appendix

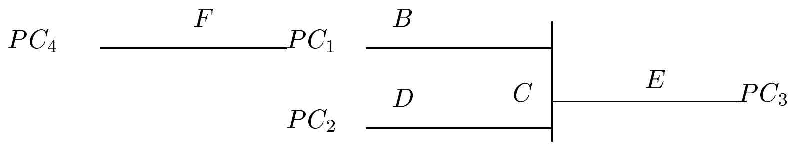

Figure 2: Adding PC_4 to the network.

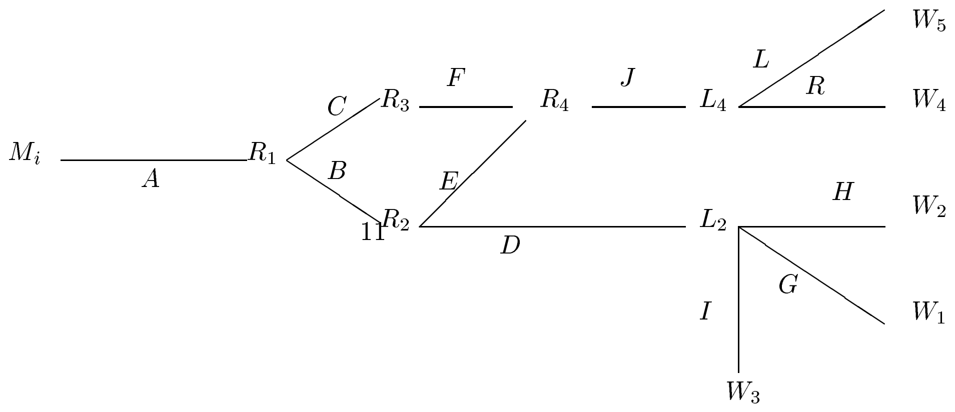

Figure 3: Part of a large network led by monitor M_i.