An Alarm Filtering Algorithm For Optical Communication Networks

C. Mas*, J.-Y. Le Boudec
Laboratoire de Reseaux de Communication
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
E-mail: {mas, leboudec}@lrc.epfl.ch

Abstract

A single failure in a communication network can trigger many alarms. We propose an alarm filtering algorithm for the management of an optical network using Wavelength Division Multiplexing (WDM). The algorithm supports: (i) multiple failures and (ii) passive network elements that do not generate alarms but may fail. This algorithm will be applied to the network of the ACTS COBNET project.

1 Introduction

In the last few years, there has been a rapid development of communication networks. Some time ago, each communication link was transmitting a single channel. By using optical fiber, several information channels can be multiplexed in one link. As a consequence, the transmitted information as well as the data rate have increased. The drawback of this evolution is that the information lost when a failure occurs has also increased.

Fault Recognition identifies the failure from the occasioned alarms and its speed is critical for ensuring availability of the network. Fault Recognition is composed of Alarm Filtering and Testing. The former gives a list of elements that may have failed and triggered all the alarms that the manager has received. Testing confirms the failure by checking each element of the Alarm Filtering result. The main part of Fault Recognition is Alarm Filtering which is essential for speeding up the testing process.

Consider an optical network using Wavelength Division Multiplexing (WDM). When a network element fails, all the channels that pass through this element are interrupted. Consequently, all the affected elements that were participating in these interrupted channels and are able to send alarms to the manager, will report an alarm. This effect is called *Alarm Storm Effect.* The manager receives all these alarms and has to find out which is the element that might have failed. In some cases, the faulty element will be able to inform the manager that it is not working properly but in other cases the element will stop working without

[*] Corresponding author

sending any information to the manager. This is particularly true for *passive* elements such as optical fibers.

The algorithm supports: (i) multiple failures and (ii) passive network elements that do not generate alarms but may fail.

In the literature there are algorithms for the localization of failures. Some of them work with probabilities of failure [1]. The problem of working with probabilities is that they are changing continuously depending on external parameters (for example age, temperature and pressure). Other algorithms use Management Information Bases to look for information related with the alarms [2]. The proposed algorithm tries to get closer to real communication networks and works with the basic data that the manager is able to get without accessing any further information.

This article is structured as follows. Section 2 introduces the network model used for the algorithm. In Section 3 the Alarm Algorithm is presented. Section 4 shows an application of this algorithm in the COBNET case. COBNET is an ACTS Project aimed at providing connections between separate Corporate Public Networks (CPNs) through the Public Network independently of the communication protocol used by the ends.

2 Model Definition

The model regarded in this article is defined as relations between network elements. A network element is a part of the network that can fail. Each network nlement is identified by two integers $e = (i, j)$. The former gives the real node where it belongs and the latter gives the component of this node. For example, the node 2 can have as a component a transceiver card that can be identified as (2.1).

There are two types of relation between network elements: *interconnection* that gives the topology of the network (subsection 2.3) and *association* that gives the established channels (subsection 2.4). In this model we consider a channel to be the establishment of a communication between two nodes.

2.1 Network Elements

This problem considers two kinds of network elements: active and passive elements. The *active* elements are able to send alarms and information to the manager (one example is a transceiver). As mentioned above, we model active elements by a couple of integers. On the drawing, they are represented by a square.

On the contrary, the *passive* elements do not give any information to the manager. One example is an optical fiber. The passive elements are modeled by a couple of integers where the first integer is 0 (to distinguish them from the active elements). On the drawing, they are represented by a circle. The identifier of this element has only one modifiable field because it is the only information that the manager has about it. For example, in figure 1, (1,2) is the second component of the node 1 and (0,1) is a passive element.

We define a as the number of active elements and n as the total number of elements in the network (active and passive).

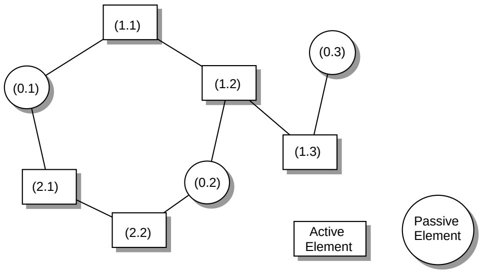

Figure 1: Interconnection Relations

2.2 Alarms

As mentioned before, active elements can send alarms to the manager. We define a *primary* alarm as the alarm resulting from the network element that has a failure and a *secondary* alarm as the alarm that is a consequence of a failure in other element.

The information given by an alarm depends on the element but typically contains [3]: (i) which element sent the alarm, (ii) when the alarm was sent, (iii) the nature of alarm and (iv) what is the reason for sending this alarm. In our analysis the information associated with an alarm contains only two pieces of data: which element has sent the alarm and what type of alarm it is because too much information can make the diagnosis of the alarms much more difficult. In particular, considering our model of an optical system, the type of alarm can be either "not-receiving" or "not-sending", referred to by the alarm identifier 1 and 2 respectively.

Note that the algorithm does not consider "wrong-functioning" alarms because when an element sends this type of alarm, the element is considered as a very probable candidate and it is checked directly. This case does not require the application of this algorithm.

The alarm structure has two fields: one with the active element identification that generated the alarm and another field with the type of alarm (1 or 2 as it has been said before). These alarms are stored by the manager in an a-dimensional vector A. An important fact that has to be considered is that the alarms do not arrive in chronological order. This is the reason why the manager accepts alarms for a certain period of time.

2.3 Interconnection Relation

We define a relation between network elements called *interconnection*. A network element with identifier e_1 is *interconnected* with an element e_2 if they are physically connected within the network. For example in figure 1, (0,1) is interconnected to (1.1) and (0.3) is not interconnected to (1.2).

This relation is symmetric and can be represented by a nxn boolean array T. This array represents the network topology. T[e_1,e_2]=1 means that e_1 and e_2 are *interconnected* and T[e_1,e_2]=0 means that they are not.

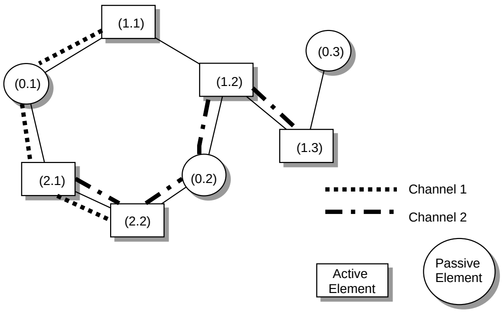

Figure 2: Association Relation with Channels

2.4 Association Relation

As in all communication networks, channels are established between nodes that want to communicate. In our model, we define a *association* as a relation between a network element and a channel. Each channel is identified by an integer. An element e_1 is *associated* with a channel c_1 if e_1 belongs to the channel c_1. For example in figure 2 the passive element (0.1) is *associated* with channel 1 and the active element (1.2) is *associated* with channel 2.
The properties of the communication relation are:

- Two or more elements can be associated with the same identifier;
- A network element can be related to 0, 1 or more channels.

This relation can also be represented in a cxn boolean array C where c is the number of established channels. C[c_1,e_1]=1 means that c_1 passes through e_1 and C[c_1,e_1]=0 means that it does not.

3 Alarm Filtering Algorithm

Assume that at certain time, the manager receives a number of alarms from the network. The goal of the Alarm Filtering Algorithm is to find the element or elements whose secondary alarms are the ones that the manager has received. The inputs to the algorithm are:

- Topology of the network T
- Established channels C
- Set of alarms received by the manager A

The output of the algorithm is a set of the most probable candidates to be faulty O. The algorithm is composed of three parts:

- Initialization: Element Domain Phase (section 3.1)
- Core:
  - Alarm Discarding Phase (section 3.2.1)

  - Candidate Generator Phase (section 3.2.2)

- Result: Comparison Phase (section 3.3)

The initialization phase has to be run every time there is a change either in the topology of the network or in the established channels. The other phases have to be applied at any change in the set of received alarms.
When the manager receives a set of alarms, he applies the Alarm Filtering Algorithm to find the best candidate(s) which with its(their) failure has(have) generated all the alarms that the manager has received. With this result, the manager testes them. If the elements turn out to be faulty, the manager acts in consequence, for example, repairing the failure.

3.1 Initialization: Element Domain Phase

This phase finds, for every network element e, the set of secondary alarms that would be generated in case it fails. In other words, if the element e fails, which alarms would the manager receive.
Inputs:

- Topology of the network T
- Established channels C

Output:

- Secondary alarms S

The nxa output array S gives, for every element, which active elements would send an alarm and which kind of alarm. As mentioned before, the considered alarms are: "not-receiving" (alarm identifier 1) and "not-sending" (alarm identifier 2).
It is important to note that this phase is independent of the alarms that are received by the manager. Once this phase has been executed, S can be kept and reused as long as there is no change either in the topology or in the established channels.
The result of this algorithm is S[e_1,a] where a are the alarms that e_1 would generate in case it fails.
The code has the following structure:

```
procedure Alarm-Domain (integer nxn array T, cxn array C)
begin
   For each established channel i do
       vector1[n]:=C[i]
       For each element of vector1 j do
           if   vector1[j]=1 then        if j-th element belongs to the i-th channel
                call Generated-by (j)    Find generated alarms when j-th element
                                         fails
                j:=j+1
           i:=i+1
   Return S[n,a]
end
```

```
procedure Generated-by (integer j)
/comment: This procedure finds which elements would send an alarm and
which kind of alarm would be in case the element j fails/
begin
    vector[2]:=vector1[n]                    Modifiable copy of the elements that
                                             belong to the same channel than the j-th
                                             element
    (a,b):=Find_next_active_neighbors (j, vector2[n])
    call evaluate_A (j,a)                    Gives the kind of alarm from a
    if b <> 0 then evaluate_A (j,b)          If there is a second neighbor, the kind
                                             of alarm is found from it
    vector2[j]:=0                            The studied element is eliminated in
                                             order to no come back to it
    call Generated-by (a)                    Find the generated alarms when a-
                                             element fails
    if b <> 0 then Generated-by (b)         If there is a second neighbor, find the
                                             generated alarms when it fails
end
```

```
procedure Find_next_active_neighbors (integer j, n-dimensional vector vector)
/comment: This procedure finds the 2 neighboring active elements of the
jth-element that belong to the channel given by vector[n]/
begin
    integer found-neighbors:=0
    while (the 2 next active neighbors have not been found) do
        x:=T[j,k]                            x is 1 if j and k are connected
        if (x=1 AND vector[k]=1) then        If j and k are connected and k
                                             belong to the same channel then
            if found-neighbors=0 then
                a:=k                         k is the first next active element
                found-neighbors:=1
            else
                b:=k                         k is the second active element
                found-neighbors:=2
        k:=k+1
    Return (a,b)
end
```

```
procedure evaluate_A (integer j,k)
/comment: This procedure gives the kind of alarm from the k-th element.
If the j-th and the k-th element are close neighbors then the alarm is a
not-sending alarm. If these elements have a passive element between
them, the alarm is a not-receiving alarm./
begin
    if T[j,k]=1 then S[j,k]:=2
    else S[j,k]:=1
end
```

3.2 Core of the Algorithm

In this section the main part of the algorithm is presented. It is composed of two parts. The former is called Alarm Discarding Phase because it eliminates obsolete alarms that do not give useful information to locate the failure. The latter is called Candidate Generator and finds the candidates for the subset of alarms resulting from the previous phase.

3.2.1 Alarm Discarding Phase

Inputs:

- Topology of the network *T*
- Established channels *C*
- Set of received alarms *A*

Output:

- Subset of received alarms *A'*

The main goal of this phase is to recognize and discard obsolete alarms. In fact, many of the alarms do not give any useful information for the localization of the failure. These alarms are called obsolete alarms because they are clear consequences of a failure and they can be related to other alarms. The non-redundant alarms are stored in a a-dimensional vector *A'* and the redundant alarms are stored in a second a-dimensional vector *Elim.*

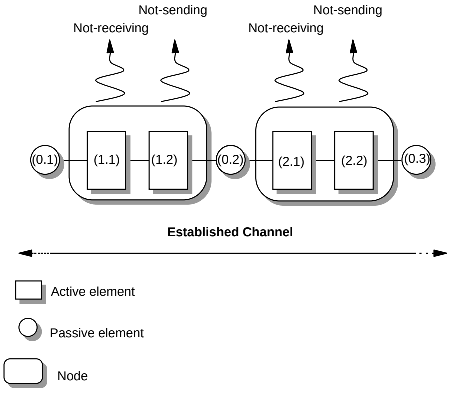

Figure 3: Alarm Discarding example

For example, in figure 3, the active elements (1.1), (1.2), (2.1) and (2.2) belong to an established bidirectional channel. (1.1) and (1.2) belong to network node 1 and (2.1) and (2.2) to network node 2. Due to a failure in the network, the element (1.1) which could be a receiver card, does not receive any signal from (0.1). As a consequence, the first node is not receiving any signal. The element (1.1) warns the manager that is not receiving and element (1.2) warns the manager that it is not sending any signal to the next element. The passive element (0.2) can not send any warning to the manager. The second node reacts in the same way than the first one. The alarms from (1.2), (2.1) and (2.2) are obsolete because they are a clear consequence of the alarm in (1.1). Therefore, the only alarm that the algorithm will keep in A' is the alarm from element (1.1) because this is not a consequence of any other alarm (in the situation described in the figure).

3.2.2 Candidate Generator Phase

Inputs:

- Topology of the network *T*
- Established channels *C*
- Subset of received alarms *A*

Output:

- Associated elements *E'*

This phase finds the elements that could have caused the alarms received by the manager. These elements will be stored in a n-dimensional vector *E'*. The procedure consists of three steps:
- The first step considers the elements that have sent the alarms. All the elements of A' are included in E'.
- The second step adds to E' the neighboring active elements of the A' elements that have not been discarded in the Alarm Discarding Phase (not included in Elim). This step is necessary to cover the case when a faulty element cannot send any alarm when it fails.
- The third step finds the passive elements, if there are any, between each pair of active elements and adds them into E'.
At the end of this phase E' contains all the candidates which could have failed.

```
procedure Candidate-Generator (integer a-dimensional array A')
begin
    For all the active elements of A'[a] i do
        E'[i]:=1                        Add the i-th element in E'[n]
        (a,b):=Find_next_active_neighbors (j, vector2[n])
        If a does not belong to Elim[a] then
            E'[a]:=1                    Add the a-element in E'[n]
            If T[i,a]=0 then            If there is any passive element between
                                        i and a-element
                b:=Passive(i,a)         b is the passive element between i and a
                E'[b]:=1                b is added in E'[n]
    Return E'[n]
end
```

```
procedure Passive (integer a,b)
/comment: This algorithm find the passive element between the two given active
elements a and b./
begin
    x:=false
    p:=(n-a)+1                          p is initialized to the first passive element
    Repeat
        If (T[a,p]=1 AND T[b,p]=1) then   If p is connected to both passive elements
           x:=true                        Finish
        else
           p:=p+1                         Otherwise continue with the next passive
                                          element
    Until is x=true
    Return p
end
```

3.3 Result: Comparison Phase

Inputs:

- Associated elements *E'*
- Secondary alarms *S*
- Set of received alarms *A*

Output:

- Probable faulty elements *O*

The elements belonging to E' are now analyzed. The analysis consists of choosing an element or several elements from E', finding the secondary alarms corresponding to them in S and comparing them with the set of received alarms A. The algorithm considers the possibility of multiple failures. Since one failure is more probable than several simultaneous failures, the selection procedure first considers single elements failing before going on considering multiple failures. With the chosen element we check its secondary alarms (given by S[e,a]) and compare them with the alarms received by the manager. In case several candidates are chosen as a set, the union of their secondary alarms are compared with the received alarms.
Once both sets are equal, the n-dimensional vector *O* is dated up to the last set of tested elements.

```
procedure Comparison (integer nxa array S, n-dimensional vector E')
begin
    x:=true          x gives when the most probable faulty element has been found
    Repeat
        i:=Choose-Candidate(E'[n])
        j:=0; t:=true
        while ($j < n$ AND x=false AND $i$ <> 0) do
           If S[i,j]<>A[j] then t:=false
           j:=j+1
        if j=n then x:=true
    Until x=true
    Return O[n]
end
```

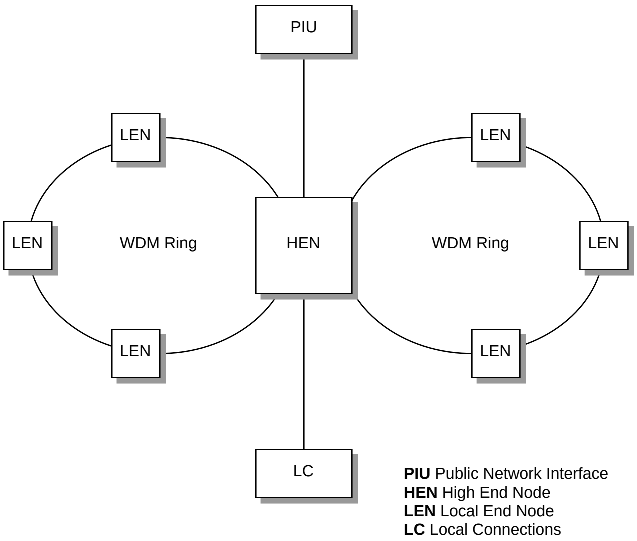

Figure 4: COBNET Network

4 Application to the COBNET case

This Alarm Filtering Algorithm is applied to Corporate Public Networks (CPNs) of the COBNET Project. CPNs are composed by a central switch located in the High End Node (HEN) that interconnects the ports from one or more rings with local ports and ports to the Public Network. Figure 4 shows a concrete case of a CPN with two rings. The CPN manager, which resides in the Telecommunication Management Network (TMN), receives the alarms from all the elements of its domain.

Suppose there are three channels established in the network as illustrated in Figure 5. The relation *association* is described in the table of figure 6. The first channel C1 goes from the Public Network to the LEN 5, the second channel C2 connects the LEN 4 with LEN 8 (local channel) and the third channel C3 connects the Public Network to LEN 9. We modeled the COBNET network as shown in figure 5. All the passive elements are also represented and nodes are divided into separate parts that are active and independent.

Suppose a set of alarms is received. First of all the manager will check if the fault is in its domain and if any alarm is a "wrong-functioning" alarm. If the failure is in the CPN and no element sends an alarm of type 3, the manager checks if there has been any change in the topology or in the established channels. If there have been changes, the initialization Element Domain phase has to be run again. Otherwise, the manager executes the main part of the algorithm. The result of the element domain phase is shown in figure 7.

Assume that during a period of time, the manager has received the set of alarms described in figure 8. The manager applies the main part of the algorithm: first discards the obsolete alarms and then finds the likely candidates:

- Alarm Discarding Phase: The alarms from elements 4.1, 2.2 and 8.1 can be considered as obsolete alarms and they are discarded. The resulting

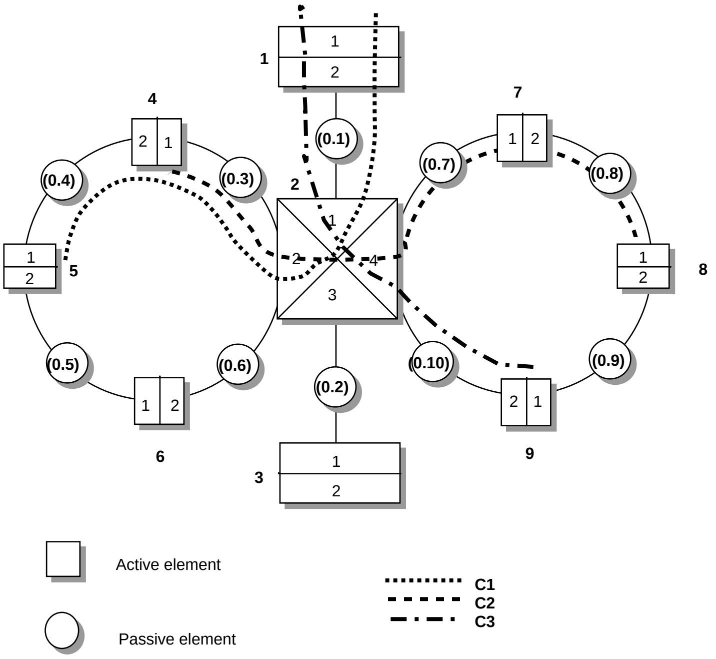

Figure 5: COBNET Network model

C	1.1	1.2	2.1	2.2	2.3	2.4	3.1	3.2	4.1	4.2	5.1	5.2	6.1	6.2	7.1	7.2	8.1	8.2	9.1	9.2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10
1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
2	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	1	0	0	0
3	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0

Figure 6: Channel array C

vector A' is in figure 9.

- Candidate Generator Phase: The element that may have caused these alarms are in shown figure 10.
  The algorithm has been able to reduce the number of candidates from 11 to 4.

- Comparison Phase: In this phase, one or several candidates are analyzed by comparing their secondary alarms with the alarm received by the manager. In this example, the element 7.1 is most probably the one that has failed because its secondary alarms are the alarms that the manager has received.

5 Conclusion

The manager receives many alarms every time a failure occurs. In order to find the failure, Fault Recognition with an effective Alarm Filtering Algorithm is needed. The network model used in this article is a simple and general one that is based on the interconnection of elements. It distinguishes between active and

	1.1	1.2	2.1	2.2	2.3	2.4	3.1	3.2	4.1	4.2	5.1	5.2	6.1	6.2	7.1	7.2	8.1	8.2	9.1	9.2
1.1	x	2	1	2	0	2	0	0	1	2	1	0	0	0	0	0	0	0	0	1
1.2	2	x	2	2	0	2	0	0	1	2	1	0	0	0	0	0	0	0	0	1
2.1	2	1	x	2	0	2	0	0	1	2	1	0	0	0	0	0	0	0	0	1
2.2	2	1	2	x	0	2	0	0	1	2	1	0	0	0	1	2	1	0	0	0
2.3	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.4	2	1	2	2	0	x	0	0	1	0	0	0	0	0	1	2	1	0	0	1
3.1	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0
4.1	2	1	2	1	0	2	0	0	x	2	1	0	0	0	1	2	1	0	0	0
4.2	2	1	2	1	0	0	0	0	2	x	1	0	0	0	0	0	0	0	0	0
5.1	2	1	2	1	0	0	0	0	2	1	x	0	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0
7.1	0	0	0	2	0	1	0	0	1	0	0	0	0	0	x	2	1	0	0	0
7.2	0	0	0	2	0	1	0	0	1	0	0	0	0	0	2	x	1	0	0	0
8.1	0	0	0	2	0	1	0	0	1	0	0	0	0	0	2	1	x	0	0	0
8.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0
9.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0
9.2	2	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	x
P.1	2	1	1	2	0	2	0	0	1	2	1	0	0	0	0	0	0	0	0	1
P.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P.3	2	1	2	1	0	2	0	0	1	2	1	0	0	0	1	2	1	0	0	0
P.4	2	1	2	1	0	0	0	0	2	1	1	0	0	0	0	0	0	0	0	0
P.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P.7	0	0	0	2	0	1	0	0	1	0	0	0	0	0	1	2	1	0	0	0
P.8	0	0	0	2	0	1	0	0	1	0	0	0	0	0	2	1	1	0	0	0
P.9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P.10	2	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 7: Secondary alarms S

1.1	1.2	2.1	2.2	2.3	2.4	3.1	3.2	4.1	4.2	5.1	5.2	6.1	6.2	7.1	7.2	8.1	8.2	9.1	9.2
0	0	0	2	0	1	0	0	1	0	0	0	0	0	0	2	1	0	0	0

Figure 8: Received alarms A

passive elements. The Alarm Filtering Algorithm locates failures occurring in both active and passive elements giving a more precise localization.
One main aspect of the algorithm is the minimal information that it needs from the alarms i.e., only the elements that originated the alarm and what type of alarm it is (2 possibilities).
We show that this algorithm is able to reduce by more than 60% the number of likely candidates that have failed. From this subset, the algorithm gives the most probable faulty element(s). Thus the time required to isolate the failure considerably and data loss in minimized. Finally, we considered a real application in the COBNET ACTS Project.

References

[1] I. Katzela and M. Schwartz, "Schemes for Fault Identification in Communication Networks," *IEEE/ACM Transactions on Networking*, vol. 3, December 1995.

[2] J. F. Jordaan and M. E. Paterok, "Event Correlation in Heterogeneus Networks using OSI Management Framework," *Integrated Network Management*, vol. III, 1993.

[3] A. T. Bouloutas and A. Finkel, "Alarm Correlation and Fault Ientification in Communication Networks," *IEEE Transactions on Communications*, vol. 42, Feb/March/April 1994.

1.1	1.2	2.1	2.2	2.3	2.4	3.1	3.2	4.1	4.2	5.1	5.2	6.1	6.2	7.1	7.2	8.1	8.2	9.1	9.2
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0

Figure 9: Resulting alarms A'

1.1	1.2	2.1	2.2	2.3	2.4	3.1	3.2	4.1	4.2	5.1	5.2	6.1	6.2	7.1	7.2	8.1	8.2	9.1	9.2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	010
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 10: Candidates E'