Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 50(64), Fascicola 1, 2005

Algorithm for Frequent Pattern Recognition in Telecommunication Alarm Logs

Petru Serafin[1]

Abstract – In telecommunication networks all the perturbations that influence the quality of telephony services must be presented to the network monitoring system by proper means that are generically called alarms. Alarms are registered in alarm logs. This paper presents a study of data mining over alarm logs in order to determine sequences of alarms that repeat themselves with a certain frequency. Such sequences of alarms constitute frequent patterns and may be of a certain interest for network monitoring systems.

Keywords: Alarm logs, pattern recognition, candidate patterns, frequent patterns.

I. INTRODUCTION

In the actual context of the development of telecommunications, the volume of information that is transported in telecommunication networks is continually increasing. Therefore, an important matter for the network monitoring system [2] is to be able to process the information with real-time constraints in order to determine the optimal functioning conditions for the network elements. Network monitoring systems are based on data acquisition of information provided within the network.

Generally, the information of notification about functional states of network elements at a given moment is called *alarms*. The information flow of all these alarms for the entire network is registered in *alarm logs*.

The architecture of most network monitoring systems is based on a modular organization. The data acquisition process consists of collecting alarms into alarm logs. The main objective of the network monitoring system is to guarantee and increase the quality of telephony services. It is important to analyze alarm logs to determine eventual faults in the supervised system.

The analysis of alarm logs [1], [4], [7], [8] aims to diagnose the functioning state of the network elements to be able to provide data for the expert system to make decisions for operating and maintenance of the telecommunication network. The important flow of alarms transmitted to the network monitoring system reveals the problem of alarm processing. Alarm processing eventually correlates alarms into relevant categories and tries to eliminate non-relevant alarms that do not influence the quality of services.

One of the methods of alarm processing is to find sequences of alarms that repeat frequently in a telecommunication alarm log. These sequences of alarms are called frequent patterns and they may be of some importance for the network monitoring system since they express a correlation between alarms that the system has to further analyze.

II. ALARM LOGS

An alarm log is defined as a list of alarm that is ordered chronologically following their moments of appearance in the network. An alarm log contains registered information about the functional state of network elements, for a given time interval. This time interval is called the *observation window* for the alarm log.

For example, in *Figure 1* it is presented an alarm log for an observation window of *15 minutes* about two network elements, `UNIT1` and `UNIT2`:

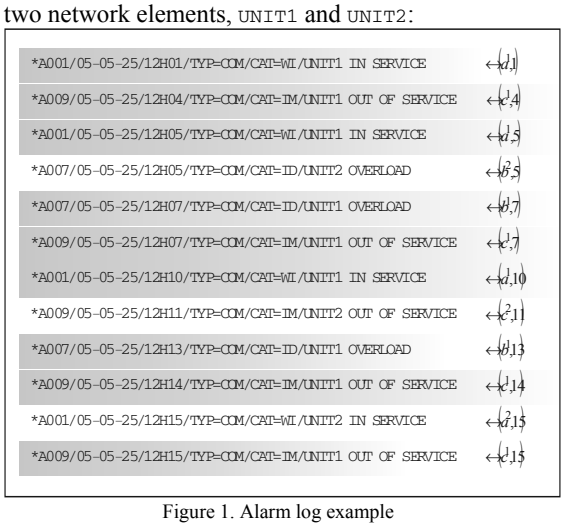

Figure 1. Alarm log example

The mathematical expression of an alarm is a couple of elements (x^i, t_0), where x^i is the type of

[1] Alcatel Romania, R&D Department, 9 Gh.Lazar, 300081 Timişoara, Romania, e-mail: petru.serafin@alcatel.ro

the alarm referring to network element i, and t_0 is the moment of appearance of the alarm in the network [3]. Using this notation, the alarm log represented in *Figure 1* can be expressed using of a set of alarms that concern UNIT1, a^1 for IN SERVICE, b^1 for OVERLOAD and c^1 for OUT OF SERVICE. In the same manner, concerning network element UNIT2, it can be expressed using the alarm set a^2, b^2 and c^2.

With the notations introduced above, the alarm log in *Figure 1* can be expressed mathematically as follows:

$$J = \{(a^1,1)(c^1,4)(a^1,5)(b^2,5)(b^1,7)(c^1,7)(a^1,10)(c^2,11)(b^1,13)(c^1,14)(a^2,15)(c^1,15)\} \quad (1)$$

An important observation is that the alarms which compose the log in equation (1) are distinct one to another, because each alarm has its own type and moment of appearance in the network. Another observation is that the alarms referring to network element UNIT1 are repetitive because they can be found more than one time in the alarm log. Therefore from the given alarm log we may be able to extract two sub-logs, each referring to a different network element.

For network element UNIT1, the mathematical expression of the given alarm log is the following:

$$J^1_{t=[1..15]} = \{(a^1,1)(c^1,4)(a^1,5)(b^1,7)(c^1,7)(a^1,10)(b^1,13)(c^1,14)(c^1,15)\} \quad (2)$$

For network element UNIT2, the alarm log can be expressed as follows:

$$J^2_{t=[1..15]} = \{(b^2,5)(c^2,11)(a^2,15)\} \quad (3)$$

In *Figure 2* it is depicted the graphical implementation of the sub-log in relation (2) for network element UNIT1:

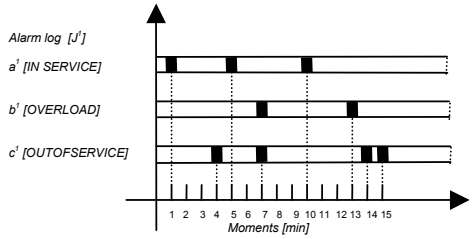

Figure 2. Moments in an alarm log

An important remark about equations (1) and (2) is that they may express some simultaneous alarms. For example $(a^1,5) \leftrightarrow (b^2,5)$, $(b^1,7) \leftrightarrow (c^1,7)$ and $(a^2,15) \leftrightarrow (c^1,15)$. In reality these alarms probably do not appear at the same time in the network but with very short delays between them. But because of the discrete timing of moments of appearance, these short delays seem as simultaneous moments. In order to keep the characteristic of simultaneous alarms in equation (2), but still not indicate the exact appearance moments, we can write the equivalent equation (4):

$$J = \left\{ a^1 c^1 a^1 \begin{matrix} b^1 \\ c^1 \end{matrix} a^1 b^1 c^1 c^1 \right\} \quad (4)$$

Equation (4) is also referred as expressing an alarm log without indicating the moments of appearance of alarms, thus meaning it does not contain temporal constraints, it contains only the ordering constraints.

III. FREQUENT PATTERNS

The main idea to analyze alarm logs to find sequences of alarms that repeat themselves with a certain frequency, is to generate some possible sequences of alarm, that are called candidate patterns, and to retain only those patterns that are frequent. This means that a calculation has to be done to determine if a candidate pattern is frequent. This calculation can be done considering the moments of appearance of each alarm in the log at the given time of the calculation or considering the moments of appearance by sub-logs.

The algorithm for frequent pattern recognition hereby presented is based on a fundamental propriety of frequent patterns (see [2]): if a pattern in an alarm log is frequent then all its sub-patterns are necessary to be also frequent in that alarm log. Another expression of this theorem is that the necessary and sufficient condition for a pattern to not be able to be frequent is that at least one of its sub-patterns is not frequent. Of course, nothing can be implied for a pattern with all its sub-patterns frequent: even if all sub-patterns are frequent we can only assume that the pattern *may be* frequent, not that it *should be* frequent.

For example, considering pattern $[abc]$ and all its sub-patterns $[a]$, $[b]$, $[c]$, $[ab]$, $[bc]$ and $[ac]$, one can clearly determine that if any of these sub-patterns is not frequent then surely the pattern $[abc]$ itself is not frequent. On the other hand, even if all sub-patterns are frequent, one can only determine that pattern $[abc]$ *may be* frequent. So, considering $[ab]$ is frequent, we determine that $[a]$ and $[b]$ are frequent. The algorithm needs to check the frequency of $[bc]$ and $[ac]$, to be able to retain $[abc]$ as a possible frequent pattern.

IV. THE ALGORITHM

The algorithm can be written as a logical diagram as presented in *Figure 3*. At each iteration i of the algorithm, there are generated sub-patterns of dimension $i+1$ starting from the frequent sub-patterns of inferior dimension. The frequency calculation eliminates sub-patterns that are not frequent and therefore eliminates the whole branch of

patterns of superior order that can be deducted starting from these sub-patterns.

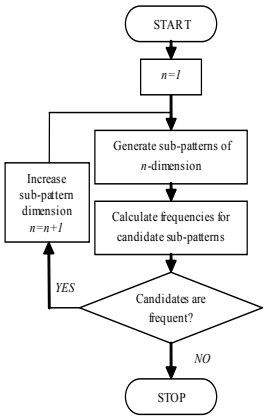

Figure 3. Logical diagram of the algorithm

An important parameter of the algorithm is the minimal considered frequency $f_{\min}$ that allows more or less complexity of the construction of candidate sub-patterns.

The algorithm ends at iteration i for which there are no more frequent patterns calculated. Of course, the maximal dimension of the solution is given by the maximal dimension of the sub-log J that is being analyzed, $n \le \dim|J|$.

Given a sub-log of alarms J and a process of pattern recognition we can note $\overline{A}_i(J)$ the set of solutions of dimension i that are found in sub-log J. Evidently the solutions depend on the strategy of sub-patterns generation and frequency calculation that are chosen. Let us suppose the strategy means that any two sub-patterns are distinct one to another.

The algorithm may be written in programming pseudo-language as follows:

Algorithm Frequent pattern recognition()
In J =alarm log or sub-log, $f_{\min}$ =minimum frequency considered
Out $\overline{A}(J)$ =assembly of frequent patterns in J log
/* Initialization of dimension 1 (unitary) alarms */

$$\overline{A}_1(J) \leftarrow \left\{ \bigcup_{\dim|a_i|=1} a_i \,|\, a_i \in J, f(a_i) \ge f_{\min} \right\};$$

/* Initialization of dimension n = 1 */
$n \leftarrow 1;$
/* Sequential do while list of frequent patterns of dimension n is not empty */
Do While $\overline{A}_n(J) \neq NULL$
/* Generate candidate sub-patterns of dimension $n+1$ */

$$A_{n+1}(J) = \left\{ \bigcup_{\dim|c_i|=n+1} c_i \,|\, c_i = candidate\_*(\overline{A}_n(J)) \right\};$$

/* Calculate minimum frequencies and retain in list only frequent sub-patterns of dimension $n+1$ */

$$\overline{A}_{n+1}(J) = \left\{ \bigcup_{\dim|a_i|=n+1} a_i \,|\, a_i \in A_{n+1}(J), f(a_i) \ge f_{\min} \right\};$$

/* Increment dimension n */
$n \leftarrow n+1;$
/* End of sequential do while */
End Do;
/* Construct list of results as assembly of all frequent sub-patterns from dimension 1 to $n-1$ form J log*/
Return $\overline{A}(J) \leftarrow \bigcup_{i=1}^{n-1} \overline{A}_i(J);$

Because we need to avoid the problem of infinite multiplicity of superior dimension candidate patterns, we must allow a predefined order between the alarms. For example we may consider the space of parallel alarms, which means candidate patterns are generated from previous dimension patterns by adding alarms of superior or equal order in the pattern. For example, the pattern *ab* generates candidate patterns *abb* and *abc*, but we need no longer to generate candidate *aab* because this pattern contains sub-pattern *aa* which is not frequent and was not retained.

The procedure for determination of frequent patterns in parallel space can be written in programming pseudo-language as follows:

Procedure *candidate parallel()* Generates parallel candidate patterns
/* Generates candidate pattern using a parallel-style space for alarm assembly */
In $\overline{A}_n(J)$ assembly of frequent patterns of dimension n
Out $A_{n+1}(J)$ assembly of candidate patterns of dimension $n+1$
/* Initialization of candidate patterns of dimension $n+1$ */
$A_{n+1}(J) = NULL;$
For $\forall a_1 a_2 ... a_i \in \overline{A}_n(J)$
For $\forall a_j \in J$ AND $a_j \ge a_k, \forall k \in [1..n]$
/* Adding alarms of superior or equal type */
Candidate-parallel-pattern = $a_1 a_2 ... a_i a_j$;
/* Verify sub-patterns of the candidate pattern */
If all sub-patterns
$a_1 a_2 ... a_i a_j \in \overline{A}_n(J)$
then $A_{n+1}(J) = A_{n+1}(J) \bigcup a_1 a_2 ... a_i a_j$;
End If;
End For;
End For;
/* Solution in candidate patterns of dimension $n+1$ */
Return $A_{n+1}(J)$;

For the determination of candidate patterns in the serial space alarms do not respect a certain order. New dimension candidate patterns are obtained just by adding alarms at the end of the assembly of previous patterns.

The procedure for determination of frequent patterns in parallel space can be written in programming pseudo-language as follows:

Procedure *candidate serial()* Generates serial candidate patterns
/* Generates candidate patterns using a serial-type assembly relation for alarms */
In $\overline{A}_n(J)$ assembly of frequent patterns of dimension n
Out $A_{n+1}(J)$ assembly of candidate patterns of order $n+1$
* Initialization of assembly of candidate patterns of order $n+1$ */
$A_{n+1}(J) = NULL;$
For $\forall a_1 a_2 ... a_i \in \overline{A}_n(J)$
For $\forall a_j \in J$
/* Adding alarms at the end of the assembly of candidate pattern */
Candidate-serial-pattern = $a_1 a_2 ... a_i a_j$;
/* Verify sub-patterns for the candidate serial pattern */
If all sub-patterns
$a_1 a_2 ... a_i a_j \in \overline{A}_n(J)$
then
$A_{n+1}(J) = A_{n+1}(J) \bigcup a_1 a_2 ... a_i a_j$;
End If;
End For;
End For;
/* Solution in candidate patterns of dimension $n+1$ */
Return $A_{n+1}(J)$;

IV. PARTIAL RESULTS

We can graphically represent the solutions for the frequent pattern recognition applied to alarm log from equation (1) in parallel-style assembly as follows:

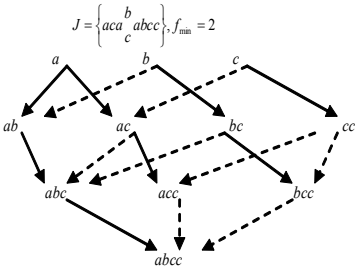

Figure 4. Frequent patterns in parallel-style assembly

Also, in serial-type assembly the frequent patterns for the same alarm log in equation 1 can be expressed as follows:

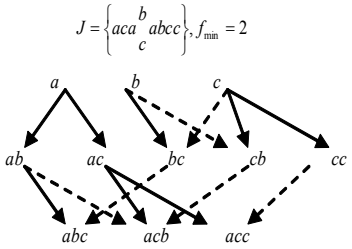

Figure 4. Frequent patterns in serial-style assembly

The software implementation of the frequent pattern recognition algorithm was realized in the development environment OMNeT++ [6] as presented in *Figure 5*:

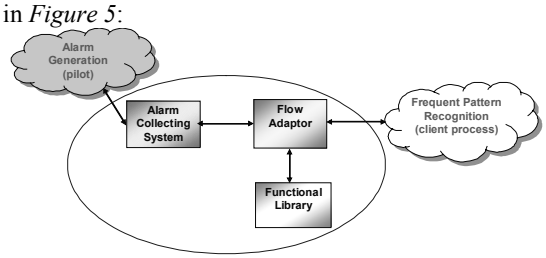

Figure 5. Module implementation

Some partial experimental results obtained in testing the algorithm over an alarm log of 10,000 alarms can be viewed in Table 1:

Minimal freq.	Frequent alarms	Candidate patterns	Frequent patterns
50	56	366	152
75	43	135	95
100	24	83	38
125	15	72	26
150	12	69	23
200	10	66	19

Table 1. Partial experimental results

For further extension of the algorithm by introducing Petri Nets formalism see [5].

IV. REFERENCES

[1] A.Aghasaryan, C.Dousson, E.Fabre, O.Osmani, Y.Pencolé – *Modeling Fault Propagation in Telecommunications Networks for Diagnosis Purposes*, Proceedings of 18th World Telecommunications Congress, Paris, 2002

[2] G.Fiche, G.Hébuterne - *Trafic et performances des réseaux de télécoms*, Ed. Hermes-Science, Groupe des Ecoles de Télécommunications & Lavoisier, Paris, 2003

[3] L.Ioan – *Probabilităţi şi variabile aleatorii în telecomunicaţii – teorie şi aplicaţii*, Ed. Matrix Rom, Bucureşti, 1998

[4] M.Mannila, H.Toivonen, A.I.Verkamo - *Discovering frequent episodes in sequences* Proceedings of 1st KDD, pp. 210-215, 1995

[5] P.Serafin – *Petri Net and Chronicle Recognition in Analysis of Telecommunication Alarm Logs*, Acta Tehnica Napocensis – Electronics and Telecommunications, Technical University of Cluj-Napoca, 2005

[6] P. Serafin – *Network Simulation using OMNet++ environment*, Buletinul Ştiinţific UPT, Tom 49 (63), Fascicola 1, pp. 407-411, Symposium of Electronics and Telecommunications, Timişoara, 22-23 October 2004

[7] Y. Pencolé, Marie-Odile Cordier - *A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks* , Artificial Intelligence Journal , Ed. Elsevier, Vol. 164, No. 1-2, p.121-170, 2005

[8] B. Guerraz, C. Dousson – *Chronicles Construction Starting from the Fault Model of the System to Diagnose*, International Workshop on Principles of Diagnosis (DX), pp.51-56, Carcassonne, France, 2004.