Dynamic Alarm Correlation based on Cellular Learning Automata in Telecommunication Networks

Mahnaz Imanazari Bonab[†], Seyed Majid Noorhosseini[††], Faezeh Akbari[†††]

[†,†††]Computer Engineering and Information Technology Department, Islamic Azad University,Qazvin, Iran
[††]Department of Computer Engineering and Information Technology , Amirkabir University of Technology, Tehran, Iran

Summary

Powerful fault management systems are increasingly required to ensure robustness and qualitative services. Though alarms are usually useful for identifying faults in such systems, huge numbers of alarms generated as a result of some major network event require efficient management methods and algorithms in order to discover the root cause in a timely manner. In this paper, we propose a robust algorithm for recognizing root cause faults in a reasonable time window by dynamically clustering alarms and events. Our algorithm is composed of three stages and uses cellular learning automaton in all stages. Simulations testify to the high efficiency of this algorithm with different parameters.

Keywords: *alarm correlation, dynamic correlation, learning automata, event correlation, clustering*

1. Introduction

Failure management has traditionally been performed by human operators. Network managers are helped in their duties by simple network status monitoring software. This matter has less accuracy nowadays due to the complexity of the new interconnected networks. Operators must be assisted in their tasks by various tools other than a simple monitoring system. By an alarm, we mean a signal which indicates a failure or malfunction of some elements in the network. An alarm is an unsolicited message from a device, typically indicating a problem in the system that requires repairmen. A single fault may produce a cascade of alarm from the affected network elements. In fact, a fault can lead to another one in a chain reaction fashion; thus, it increases innumerous alarms and masks the really important ones [1]. An event is a set of correlated alarms according to a specific fault. But a fault can lead to the global event composed of different alarms, or several local events. An event can be composed of several identical alarms. Moreover, a fault can also lead to other faults. For each alarm there is one and only one associated event. Alarms are created in three kinds: periodic alarms, which are triggered at a regular period, aperiodic alarms, or alarms which do not seem to be triggered with a specific period, and finally, alarms which are triggered only once. These alarms can either be classified as periodic (with an unknown period) or aperiodic [2]. The telecommunication network management system is responsible for the recording of the alarms generated by the nodes or components in the network, and appears them to the operator. Operators of network can be expected to see about millions of alarms in the management centre.

Many algorithms and methods have been offered for management of alarms. For example, Behavioral Proximity (BP) algorithm [3] reduces the number of alarms by clustering them to their behavior and the other algorithm is Topographical Proximity (TP) [4] that exploits topographical information in the alarm data. CUFRES algorithm [1] and other algorithms have been offered based on data mining [5][6] and probability[7][8]. Currently, many systems employ event correlation engines to find fault. The problem of an automatic identification of events for correlation purposes has been tackled from various perspectives. Model traversal approaches aim to represent the interrelations between the components of the network or the causal relations between the possible events in the network or a combination of the two. Correlations are identified as alarms propagate through the model. Rule-based [9] and Code-based [10] systems also show the relations between the events in the system, which specifies correlations according to a rule-set or codebook. Other AI techniques, such as neural networks [11], [12] or decision-trees, have also been applied to the task. Generated Alarms in a network can create events and clustering of events that show faults in a system. Types of alarms vary from one system to another and may be divided into four main categories [6][2]:

1. Hardware failure
2. Software failure
3. Functional failure (perceived external fault or feature)
4. Environmental failure (e.g. fire, temperature, etc.)

In fact, most of alarms do not contain the information about the root cause of a fault. When a fault occurs in the networks, it may incur many alarms. So, some alarms are redundant, which make fault processing more difficult [13]. Here we offer two definitions for alarm [13]:

Manuscript received December 5, 2009
Manuscript revised December 20, 2009

A. Definition 1. An alarm event

An alarm event is defined as $E_i=<e_i,t_n>$, i, n=1,2,3,., where e_i is an alarm type and t_n is its time of occurance**.**

B. Definition2. An alarm type

An alarm type is defined as $e_i=<$object_class,object-instance,alarm-num,desc$>$, i=1,2,3,., where object-class is the serial NO. of object class, object-instance is the serial NO. of object instance, alarm-number is the NO. of alarm type and desc is the alarm information consisting of alarm priority and alarm description. When alarms have the same object-class and object-instance, they cab be classified in same categories.

The paper examines other subjects as follows. In Section 2, the learning automata are briefly reviewed, Section 3 discusses the motion cellular learning automata and section 4 illustrates the clustering algorithm based on the motion cellular learning automata model. The proposed algorithm is presented in Section 5. Section 6 gives the simulation results and finally section 7, being the conclusion, will includes the results and suggestions.

2. Learning Automata (LA)

A learning automaton is a decision-making device that is able to do finite actions. In each step, a learning automaton chooses one action from action-set. The environment evaluates selected actions and automata use environment answers to choose next action and updates the internal structure. Automata learn to choose the best action from action-set [14]. In fact, the objective for the automaton is to identify the optimal action. Figure 1 depicts the relationship between an automaton and its environment.

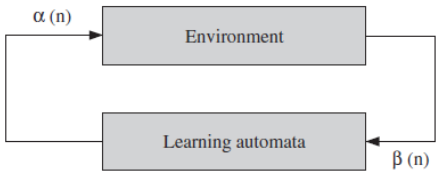

Fig. 1 Relationship between learning automata and its environment

Environment can be defined by the triple $E=\{\alpha,\beta,c\}$ where $\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_r\}$ represents a finite input set, $\beta=\{\beta_1,\beta_2,\ldots,\beta_r\}$ represents the output set, and $c = \{c_1,c_2,\ldots,c_r\}$ is a set of penalty probabilities, where each element c_i of c corresponds to one input action a_i.Environments in which β can take only binary values 0 or 1 are referred to as P-models. Further generalization of the environment allows finite output sets with more than two elements that take values in the interval [0, 1]. Such an environment is referred to as Q-model. Finally, when the output of the environment is a continuous random variable which assumes values in the interval [0, 1], it is referred as an S-model. Learning automata are classified into fixed-structure stochastic and variable-structure stochastic. In the following, we consider only variable-structure automaton. A variable-structure automaton is defined by the quadruple $\{\alpha,\beta,p,T\}$ in which $\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_r\}$ represents the action set of the automaton, $\beta=\{\beta_1,\beta_2,\ldots,\beta_r\}$ represents the input set, $p=\{p_1,p_2,\ldots,p_r\}$ represents the action probability set, and finally $p(n+1) = T[\alpha(n),\beta(n),p(n)]$ represents the learning algorithm. This automaton operates as follows. Based on the action probability set p, the automaton randomly selects an action α_i, and performs it on the environment. After receiving the environment's reinforcement signal, the automaton updates its action probability set based on Eq. (1), for favourite responses, and Eq.(2) for unfavourite ones.

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)\,p_j(n) \qquad (1)$$

$$p_i(n+1) = (1-b)\,p_i(n) \qquad (2)$$
$$p_j(n+1) = (b/r-1) + (1+b)\,p_j(n)$$

In these two equations, a and b are received reward and penalty parameters respectively. S-model, automaton updates its action probability set based on Eq.(3) and Eq.(4) .

$$p_i(n+1) = p_i(n) - \beta(n)\,b.p_i(n) + (1-\beta(n))\,a.[1-p_i(n)] \qquad (3),(4)$$
$$p_j(n+1) = p_j(n) + \beta(n)\left(\frac{b}{r-1} + (-b)p_j(n)\right) - (1-\beta(n))\,a(1-p_j(n))$$

where $\beta(n)$ is an environment's reinforcement signal in the interval [0,1]. For a=b, learning algorithm is called L_{R-P} , for a<b, it is called L_{R-P}, and for b = 0, it is called L_{R-I} .

A cellular learning automaton is grid cellular that every cell can have k states. A cellular automaton is composed of component-set that every component behavior is modified and is specified based on last experiences and neighbours behavior [15]. Every cell includes one automaton with finite states. In one-dimension, every cell has two close neighbours. Neighbourhood in one-dimension learning automaton can be extended to involve more than two neighbours. Automata can assume radius r for neighbourhood. Of course, automata assumes nearest neighbours. Cellular automata cells can be set in various dimension grids where definitions and laws of neighbourhood have been altered by the proportion of dimension. A d-dimension cellular learning automaton is a structure $CLA=(z^d,\phi,A,N,F)$, where z^d is a lattice of d-tupple of integer numbers , ϕ is a finite set of states, A is the set of LAs each of which is assigned to one cell of the CLA , $N=\{\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_{\overline{m}}\}$ is a finite subset of Z^d

called neighbourhood vector, $F : \underline{\phi}^{\overline{m}} \rightarrow \underline{\beta}$ is the local rule of the cellular learning automaton where β is the set of values that the reinforcement signal can take [16]. The General general learning automaton is a two-dimensional cellular learning automaton. Important types of neighbourhoods are Moore, Cole and Smith. In Moore neighbourhood for every central cell there are eight neighbour cells.

3. Motion Cellular Learning Automata (M-CLA)

In motion cellular learning automaton, learning automata are independent of cells [17]. Cell includes one learning automaton or more than one learning automata, called active cell. Like cellular learning automata, cellular structure and local law is impressive in motion cellular learning automaton to update the internal structure of learning automaton that resides in the active cell. Only active neighbour cells next to an active cell affect the optimized action learning of every learning automaton. Instructions of the motion cellular learning automaton are illustrated here. In every moment, every learning automaton in cellular learning automata chooses one action from its action-set based on the action probability vector. Selected action causes movement of the learning automaton from one cell to another. If pervious cell has contained learning automaton, then it does not contain any learning automaton, called inactive cell. Learning automata actions in every active cell receive reward or penalty based on current learning automata actions in neighbouring active cells and current local laws. Receiving reward and penalty leads to the updating of internal structure of learning automaton in active cell. Rewarding or giving penalties continues until the system arrives to a permanent state or former criterion is existed in automaton. Updating action of automaton structure is performed concurrently by learning algorithm in cellular learning automaton.

4. Clustering algorithm based on motion cellular learning automata model

For every n-dimension data is assigned learning automata, is called an agent [17]. Allowable neighbourhood of agent for moving is called movement direction. Two similar agents tend to move in a specific direction and this similarity of movement direction is called similar-side direction. Two different agents tend to move in different directions and this difference of movement direction is called dissimilar-side direction. In data clustering, every agent moves to a direction based on its movement direction probability vector. If the similarity adjustment between current agent and neighbour agent is high, the agent changes its movement direction based on similarity adjustment to the neighbouring agent direction. In other words, current agent learns movement direction of neighbour agent and this movement direction is the suitable direction to find similar agents. Also if the similarity adjustment between the current agent and neighbour agent is low, the current agent changes its movement direction. In other words, current agent's learning the movement direction of the neighbouring agent is not a suitable direction to find similar agent. Dissimilar-side or similar-side direction of agent for finding similar agent leads to learning of favorite movement direction based on local information. Similarity adjustment is the outcome of a similarity equation. In a motion-learning cellular automaton, every learning automaton is equal with one agent and that agent includes an n-dimension data. Every cell includes one agent, called active cell. Also it is possible to say in every cell there is more than one agent. A cellular automaton is assumed an environment for an agent's movement. Every agent chooses a movement direction based on defined directions and its neighbourhoods and probability of different directions. After movement, if the current agent has neighbouring agents, the current agent updates its probability vector. This updating is performed based on reward and penalty that the current agent acquires through similarity adjustment between current agent and its neighbour agent. Two laws are assumed for clustering as follows:

1) Every neighbour agent separately affects the movement direction of the current agent.

2) The similarity between two agents is calculated based on Euclidean or cosine distance. If the similarity between two agents is high, the current agent is rewarded and will have a similar-side direction with the neighbouring agent. If the similarity between the two agents is low, the current agent receives penalty and will have a dissimilar-side direction with the neighbouring agent.

5. Proposed Algorithm

An alarm is an unsolicited message from a device, generally indicating a problem in the system that requires attention. A single fault may produce a cascade of alarms from the affected network elements. Generated alarms form events in an environment and events clustering show the existence of fault in the network. The proposed algorithm has three stages: in the first stage, generated alarm types in cells receive reward or penalty. Every generated alarm type is given reward and other alarm types are given penalty in the same and neighbour cells. In the second stage, primary events are found by captured rewards in the first stage and other parameters like time and degree. In the third stage, final event clustering is performed by motion learning automaton to find whether fault exists in the network. Every Telecommunication

node is assumed as a cell. Every cell includes an automaton. We assume K as the number of alarm types in a cell. Every node is activated by receiving alarm and the automaton chooses the suitable action. Then it gives penalty/reward to the alarm by Eq.(5) and Eq.(6). The aggregation of probabilities in a cell should be equal to 1/r that enables us to use motion learning automaton in the next stage. A is a generated alarm and S is an alarm collection except the generated alarm in cell. When an alarm is generated, learning automaton gives reward or penalty in the same and neighbour nodes based on the Moore method.

$$p_{n+1}(A) = p_n(A) + \alpha \times \frac{1}{r} - \alpha \times p_n(A)) \qquad (5),(6)$$

$$p_{n+1}(S) = (1-b)\, p_n(S) \qquad \forall s \in S \Rightarrow s \neq A, |\, S \,| \leq k$$

Where α is a reward parameter and b is a penalty parameter, r is the number of telecommunication nodes. When an alarm type is generated vastly, the alarm probabilities increase immediately. When an alarm is generated, rewards and penalties are updated in a cell and its neighbours. Now, the automaton should find correlations between the alarm of cell and its neighbours. We can find faults in system by finding the alarm correlation between alarms and we can fix it. After rewarding or giving penalty, every automaton in cells calculates Eq. (7) for all available events in parallel. All automata acquire value of AVL(E) and this value is equal.

(7)

$$\mathrm{AVL(E)} = \frac{(\sum_{i=0}^{n} \mathrm{R(e_i)} + \sum_{i=0}^{n} \mathrm{degree(e_i)} + \sum_{i=0}^{n} \mathrm{T(e_i)}) \times \gamma}{\mathrm{AlarmNumbre}}$$

Where $R(e)$ is the value of penalties and rewards that the automaton has given to the alarm type e . As we mentioned before, alarms are divided into four types: hardware and software, functional and environmental. Hardware alarms are important because these alarms can show the fault core. Functional alarms are conversely unimportant because these alarms arisen from hardware or software faults. $Degree$ shows the significance of every alarm type. In hardware and software alarms, degree is valuable. $T(e)$ shows the interval between the starting point of an alarm type until the automaton starts to calculate the AVL(E) . γ parameter specifies the resolution to calculate the primary events. If γ value is high, then the primary events would be found with upside accuracy. γ causes different number of faults in the network. In fact, we will not have fixed fault number for different networks. We can find correct fault number by changing γ value in the network. Then, every automaton calculates Eq.(8) for every generated event in its cell. If Eq.(8) is indefeasible on some events, then these events are called primary events and If Eq.(8) is not indefeasible on some events, then these events are called secondary events:

$$R(e_i) + \deg ree(e_i) + T(e_i) > AVL(E) \qquad (8)$$

Primary event numbers impact the final cluster number. In fact, primary events show final cluster numbers and final clusters show faults in the system. Every cell including the primary event is an active cell and a motion-learning automaton starts to create its own cluster. In proposed algorithm, we use geographical distance instead of Euclidean or Cosine distance. Reward and penalty values of motion learning automaton have been showed in simulation results.

6. Simulation Results

In this section, we conduct simulations to evaluate our proposed method. We perform simulations with different nodes. Alarms are generated absolutely random as the location and position of nodes are. We perform simulations with different alarms, nodes numbers and γ values. We assume four alarm types in different simulations. Alarm types labeled with {A,B,C,D} and in it A shows hardware alarms, B shows software alarms, C shows conditional alarms and D shows environmental alarms. We assume that an alarm *A*, *B*, *C*, or *D* can appear in any node of the network. If we calculate the alarm correlation for every received alarm then the efficiency of the system would be decreased because one new alarm can not probably creates new correlations in the system. We try to find fault in the interval time or when operator requests results. In the first simulation, we assume 25 nodes and 2000 generated alarms and different γ in figure (2). Primary events vary with diverse values of γ .

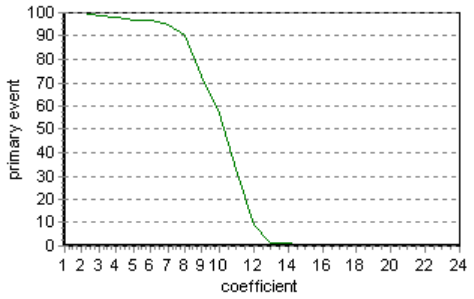

Fig. 2 Primary event numbers proportion with γ values

We repeated the simulation with different Degrees. In the next experiment, we assume value less than one for Degrees. The degree value for A is 1, for B is 0.7 and for C is 0.3, and Degree value for D is 0.1 with 100 nodes and 7000 alarms. We choose different degrees that these are

used in Eq.(7). When an alarm is created, degree value is assigned to it based on its type.

Red color shows that how many hardware events are primary events in figure (3). When we choose high coefficient, it shows just hardware events. Coefficient from 1 to 14 shows four type events that can be primary and Coefficient from 17.5 to 20 shows hardware events in figure (3).

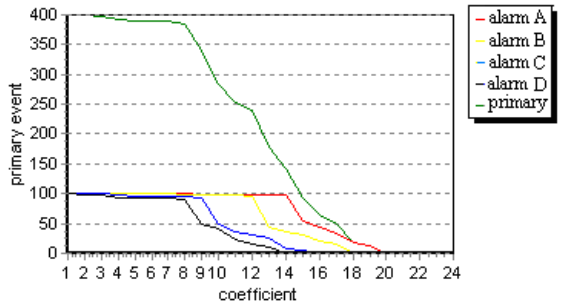

Fig. 3 Different alarm types

Experiment shows various primary events with different Degrees for different alarm types. The best result has been acquired as stair diagram in which the Degree values of alarm types are calculated as follows.

Degree A = n

Degree B = 3/4n

Degree C = 2/4n

Degree D = 1/4n

Where n is the node numbers in telecommunication network. In the next experiment, the degree value for A is 100, for B is 75, and for C it is 50, and the degree value for D is 25 with 100 nodes and 10000 alarms.

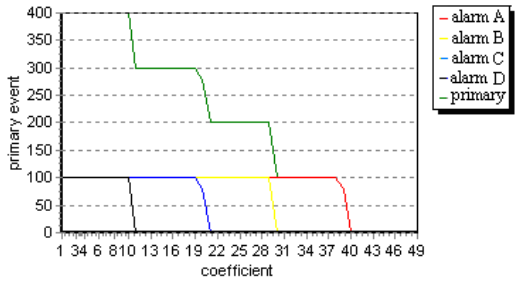

Fig. 4 primary event numbers with evaluated Degrees

Changing the degrees and γ values leads to change in primary events. The diversity of primary event numbers directly affects the final fault. Here, we show primary event with different Degrees. The blue color shows primary events with similar Degrees.

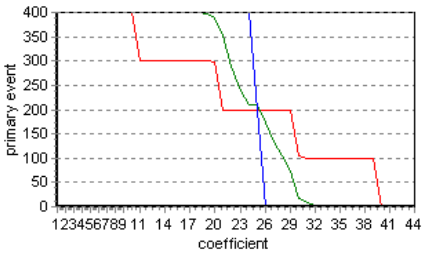

Fig. 5 Different primary events with 10000 alarms and 100 nodes

After finding the primary events, we should find the final cluster or the final fault. In this stage, experiments are repeated for γ=18 by motion learning automaton. After the second stage, every cell that has primary event is an active cell and it starts to create its cluster. In motion learning automaton is used S-model and $a = 1e-5$, $b = 1e-1$. Where a,b are reward and penalty value respectively. Alarms in clusters are different. Alarms in cluster show the number of alarms that the network has generated due to a fault.

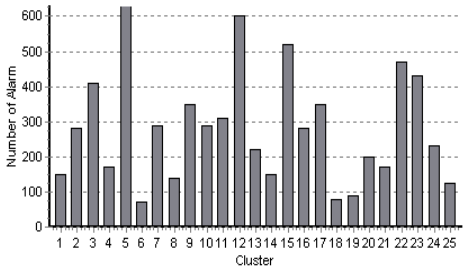

Fig. 6 number of alarms in every cluster with 8000 alarms and 100 nodes

This algorithm has only one problem that appears rarely. This problem occurs when a primary event that has occurred once in the network is lost. Of course, this problem occurs rarely. We can design algorithms by learning machine and other events can recognize its lost primary event.

7. Conclusion

In this paper, we have addressed the problem of finding the root cause of alarms in an alarm flooding situation. The proposed method introduced in this paper uses learning automaton dynamically to cluster the incoming alarms and find the root cause. Our experiments in a simulated environment show that our method is accurate 90% of the time.

References

[1] Jacques-H.Bellec,M-Tahar Kechadi, "*CUFRES: Clustering Using Fuzzy Representative Events Selection for the Fault Recognition Problem in Telecommunication Network*",Lisboa ,Portugal , ACM , 2007

[2] Jacques-H.Bellec,M-Tahar Kechadi, Joe Carthy, *"Performance Evaluation of Two Data Mining Techniques of Network Alarms Analysis"*,Conference On Data mining,Las Vegas ,NV,USA,june 2006

[3] Bellec,J.H., Kechadi , M.T., J.Carthy,*"A New Efficient Clustering Algorithm for Network Alarm Analysis*" ,Coference on Parallel and Distributed Computed Computing and System, AZ ,USA , Nov 14-16 2005

[4] A.Devit, J.Duffin, and R.Moloney,*"topographical Proximity for Mining Network Alarm Data"* , USA ,ACM ,aug 22-26 2005

[5] Klause Julisch ,*"Mining Alarm Clusters to Improve Alarm Handling Efficiency*" ,Zurich Research Laboratory ,2002

[6] Robert D.Gardner , David A.Harle, *"Fault Resolution and Alarm Correlation in High-Speed Network Using Database Mining Techniques"*, Singapor,9-12 September 1997

[7] Okuthe P. Kogeda, Johnson I. Agbinya and Christian W. Omlin,*" A Probabilistic Approach To Faults Prediction in Cellular Networks*", ICNICONSMCL'06 , IEEE,2006

[8] Zhen Guo, Guofei Jiang, Haifeng Chen, Kenji Yoshihira,*" Tracking Probabilistic Correlation of Monitoring Data for Fault Detection in Complex Systems*", Conference on Dependable Systems and Networks, IEEE,2006

[9] G.Liu, A.Mok,E.Yang,*"Composite Events for Network Event Correlation*",Boston, USA, 1999,pp.247-260

[10] S.Yemini, S.Kliger, E.Mozes,Y.Yemini and D.Ohsie , *"High Speed and Roubust Event Correlation"* , IEEE , 1996

[11] R.Gardner and D.Harle,"*Alarm Correlation and Network Fault Resolution Using Kohonen Self-Organising Map*",IEEE Global Telecom Conf , NewYork , NY ,USA ,1997

[12] H.Wietgrerf, K.-D.Tuchs, K.jomann,G.carls, P.Frohlich ,W.Nejdle and S.Steinfeld,"*Using Neural Network for Alarm Correlation in Cellular Phone Network*" Proc. Of the International Workshop on Application of Neural Network to Telecommunication 1997

[13] Qingguo Zheng,Ke Xu, Weifeng Lv,Shilong Ma,*"Intelligent Search of Correlation Alarms from Database Containing Noise Data*",Beijing 100083,china,2002

[14] Thathachar, M.A.L. and Sastry, P.S.; *"Varieties of Learning Automata: An Overview*", IEEE Transaction on Systems, and Cybernetics-Part B: Cybernetics Vol. 32, No. 6, pp. 711-722, 2002.

[15] Beigy, H. and Meybodi, M.R.; "*A Mathematical Framework for Cellular Learning Automata*", Advances in Complex Systems, Vol. 7, Nos. 3-4, pp. 295-320, September/December 2004.

[16] Hamid Beigy, Mohammad Reza Meybodi,*" Open Synchronous Cellular Learning Automata*", The CSI Journal on Computer Science and Engineering,2003

[17] Meisam Hosseini Sedehi, Mohammad Reza Meybodi," *A Data Clustering Algorithm based on Cellular Learning Automata*", Amir Kabir University,Tehran , Iran , IDMC'07, 20-21 Nov.2007 [Digests 13th Annual Conf. Comp Iran]

Mahnaz Imanazari Bonab received her B.E. degree from Payamnoor University in 2006. She received her M.E. degree from Qazvin Azad University in 2009. Her research interests include network management, databases and image processing.

Seyed Majid Noorhosseini received the B.Sc. and M.Sc degrees from Amirkabir university of technology in 1986 and 1989 , respectively. He received his PhD degree from McGill University in Montreal ,Canada in 1994. He was a senior scientist at Nortel Networks in Canada and U.S during 1996-2005, working in different areas of network management. He has a US patent 6707795 in alarm correlation method and system. He is now with the Department of Computer Engineering and Information Technology at Amirkabir University of Technology.

Faezeh Akbari received B.E. degree from Kashan Azad University in 2004. She received M.E. degree from Qazvin Azad University in 2009. Her research interests include network management as well as sensor and ad hoc networks.