BARD: A Novel Application of Bayesian Reasoning for Proactive Network Management

A. Bashar, G. P. Parr, S. I. McClean, B. W. Scotney and D. Nauck*

India-UK Advanced Technology Centre of Excellence in NGN
School of Computing and Information Engineering, University of Ulster, Coleraine BT52 1SA, UK
Email: {bashar-a, gp.parr, si.mcclean, bw.scotney}@ulster.ac.uk
*Intelligent Systems Research Centre, BT Group, Adastral Park, Ipswich IP5 3RE, UK
Email: detlef.nauck@bt.com

***Abstract*—In the context of next generation networks (NGN), there is a critical need to address the challenges facing the network management functions, to administer the converged infrastructure and services. One of the challenges is to model the unknown and probabilistic dependency relationships among the diverse network elements and adapt this model to capture the real-time network behavior. This paper proposes BARD (BAyesian Reasoner and Decision-maker), a proactive system to enhance the network performance management functions by use of machine learning technique called Bayesian Belief Networks (BBN). It exploits the predictive and diagnostic reasoning features of BBN to make accurate decisions for effective management. A case study is presented to demonstrate the application of this technique to the problem of Call Admission Control in a typical communication network. The simulation results provide the proof that BARD is an effective and feasible solution when applied for network management tasks, especially Quality-of-Service (QoS) management.**

***Index Terms*— Network Management, Next Generation Networks (NGN), Bayesian Belief Networks (BBN), Machine Learning.**

I. INTRODUCTION

THE area of communication networks has experienced a remarkable phenomenon of convergence in recent years. The possibility of providing quadruple services (voice, video, data and mobility) by bringing together the fixed and mobile network infrastructure and creation of novel applications has led to the emergence of the Next Generation Network (NGN). The International Telecommunication Union (ITU-T) has defined the NGN to be "a packet-based network able to provide telecommunication services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service related functions are independent from underlying transport-related technologies" [1]. From the architectural point of view, the core network of NGN involves a consolidation of several (dedicated or overlay) transport networks, each built for a different service, into one core transport network (based on IP and Ethernet). The access network is a highly complex mix of wired and wireless technologies, not to mention the wide variety of end terminal devices desiring to use the converged services running over the NGN. In summary, NGN promises converged services, end to end QoS, interworking with legacy networks and generalized mobility.

Challenges are abound when it comes to managing a converged network with the characteristics stated above [1] [2]. In this paper, we address the challenge of adaptively capturing the stochastic behavior of the rapidly changing network state. With the objective of automating the system monitoring functions related to network performance, we seek the help of machine learning techniques, which have the ability to learn the system behavior from past data and estimate the future behavior based on the learned system model. The vast amount of network data, collected through network management protocols (e.g. SNMP) can be utilized to construct such models and then inferences can be made regarding future states. The intelligence extracted from these inferences will be made available to an autonomous system or the human operator to initiate appropriate actions in terms of configuration changes to the network devices.

The remainder of this paper is structured as follows. In section II we provide the required background and survey some related work in this research domain. Section III describes the theory behind Bayesian Belief Networks (BBN) and the learning mechanisms which can be utilized in our solution. In section IV we present the architecture of our scheme called BARD (BAyesian Reasoner and Decision-maker) which employs the BBN learning technique. We then demonstrate the use of BARD through the setup of Opnet and Hugin simulations in Section V. The results of the simulation are presented in section VI. Section VII concludes the paper by suggesting possible future work.

II. BACKGROUND AND RELATED WORK

Network Management Systems (NMS) are used to maintain

ISBN: 978-1-902560-22-9

the network infrastructure, to assure smooth running of services, to control the operational costs of the network and to provide increased revenues to the service provider. Several standard NMS like the SNMP [3], CMIP [4] and FCAPS [5] are used to manage the telecommunication networks. Being a converged network, NGN would have to inherit some functionality from each of these systems.

The network management data sets collected through these NMS can become very large and a human operator will find it difficult to analyze and understand the unknown relationships among them. Machine learning techniques have been used to automate such tasks which use complex reasoning mechanisms to come up with patterns which are useful for network management.

Various machine learning approaches have been proposed to address the issues related to network management tasks. Decision trees have been used to achieve proactive network management by processing the data obtained from the SNMP MIB (Management Information base) objects [6]. Fuzzy logic has been applied to facilitate the task of designing a bandwidth broker [7]. Bayesian Belief Networks (BBN) have been used for providing proactive network fault detection in a telecommunication network [8]. Dynamic Bayesian Networks (DBNs) have been used to model the communication network, for detecting faults in real-time [9]. Intelligent software agents have used Bayesian reasoning to achieve network fault management [10]. Classical Recursive Least Squares (RLS) based learning and prediction was applied for achieving proactive and efficient IP resource management [11]. Reinforcement learning has been used to provide efficient bandwidth provisioning for per hop behaviour aggregates in diffserv networks [12].

Based on this survey, it can be inferred that the majority of the research work has concentrated on the fault management function of the broader network management domain. Also, it is seen that Bayesian Belief Networks are suitable for modelling and studying highly dynamic systems (e.g. the NGN). We now consider the application of BBN to achieve network performance management, which is one of the key functions of management in the FCAPS model of ITU-T [5]. Therefore, the original contribution of this work is the application of BBN technique to address the issues of performance management (in particular QoS management through Call Admission Control).

III. Bayesian Belief Networks (BBN) Theory

A. Graphical structure of BBN

A BBN is a graphical structure that allows us to represent and reason about an uncertain domain. For a set of variables $X = (X_1,....,X_n)$, a Bayesian network consists of a network structure S that encodes a set of conditional independence assertions about variables in X, and a set P of local probability distributions associated with each variable

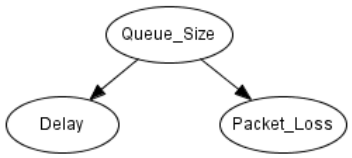

Fig. 1. An example of BBN.

[24]. An example BBN, representing the effects (*Delay* and *Packet_Loss*) of *Queue_Size* at a particular router in a network, is shown in Fig. 1. To quantify the strength of relationships among the random variables, conditional probability tables (CPTs) are associated with each node. For a typical node A, with parents $B_1, B_2,....B_n$, there is attached a CPT as given by $P(A \mid B_1, B_2,....B_n)$. For root nodes, the CPT reduces to prior probabilities.

The main principle on which BBN work, is Bayes' rule,

$$P(H \mid e) = \frac{P(e \mid H)P(H)}{P(e)} \quad (1)$$

where $P(H)$ is the prior belief about a hypothesis, $P(e \mid H)$ is the likelihood that evidence e results given H, and $P(H \mid e)$ is the posterior belief in the light of evidence e. This implies that belief concerning a given hypothesis is updated on observing some evidence.

B. Structural Learning

If the subject matter expert (or the domain modeller) knows the behaviour of all nodes and their inter-relationships, then he can build the structure manually. If the domain knowledge is limited, then structural learning algorithms use batch learning process to learn the structure. Two algorithms which are available for structure learning are - PC [13] and Necessary Path Condition (NPC) algorithms [14]. The basic idea of these constraint-based algorithms is to derive a set of conditional independence and dependence statements (CIDs) by statistical tests among the nodes of the BBN.

C. Parameter Learning

The CPTs of the nodes of the BBN are called the parameters. Again, the CPTs can be specified based on the knowledge of the domain expert, by the process of parameter elicitation. Another way is to use the past data as the basis for learning the parameters through algorithms. One such algorithm known as the Expectation Maximization (EM) is particularly useful for parametric learning [15]. The EM algorithm is an iterative algorithm which maximizes the log-likelihood score (the logarithm of the probability of the case data given the current joint probability distribution) in each iteration.

D. Sequential Learning

In dynamic systems, it is to be expected that the modeled domain changes over time. In order for the modeled domain to reflect the real domain behavior, the parameters of the model need to be updated based on the observations made. This process is termed as sequential learning. The algorithm to

implement this learning is called adaptation algorithm [16].

E. *Influence Diagrams (ID)*

To incorporate decision making capabilities, the BBN is converted to an influence diagram (ID), by adding decision nodes and utility nodes. A decision node represents the decision being made at a particular point in time. The values taken by the decision nodes are the actions which must be chosen by the decision maker. A utility node quantifies the *usefulness* of the outcomes resulting from the actions of decision. To achieve this, the utility node assumes a utility table with one entry for each possible instantiation of its parents. In our example, we have added a decision node (*Raise_Alarm*) which decides either to indicate *Safe* or *Danger* level of packet drops by the router. Also, a utility node (*Utility*) is added, which will have a utility table with 4 entries (indicating *usefulness* of a particular decision) representing the combinations of *Raise_Alarm* and *Packet_Loss* states. The resulting ID is shown in Fig. 2.

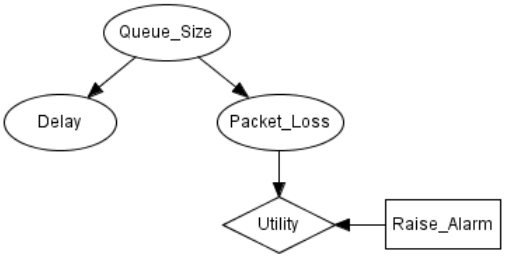

Fig. 2. Influence diagram for the BBN of Fig. 1.

F. *Belief Updating*

Evidence (observation of a node state) on a particular node is used to update the beliefs (posterior probabilities) of other nodes of the BBN. The BBN framework supports two types of reasoning: predictive and diagnostic reasoning. In predictive reasoning the causal nodes act as evidence nodes and the beliefs of the effect nodes are updated. Whereas in diagnostic reasoning the effect nodes become the evidence nodes and the beliefs of causal nodes are updated. Depending on the type of the evidence available and specific problem being addressed, either one or both reasoning may be applied. There are various algorithms (e.g. Kim and Pearl's Message Passing Algorithm [17], Lauritzen Speigelhalter Algorithm [21]) to carry out inference in BBN.

IV. Bayesian Reasoner and Decision-maker (BARD)

The domain under consideration for our proposed work is shown in Fig. 3. The network is divided into access and core networks with ingress and egress routers acting as a bridge between these distinct areas. We assume that all the network elements are SNMP-enabled and they have SNMP clients installed in them for sending MIB data. Network data from all the network elements is collected by the SNMP Manager (either by polling process or by traps sent from clients).

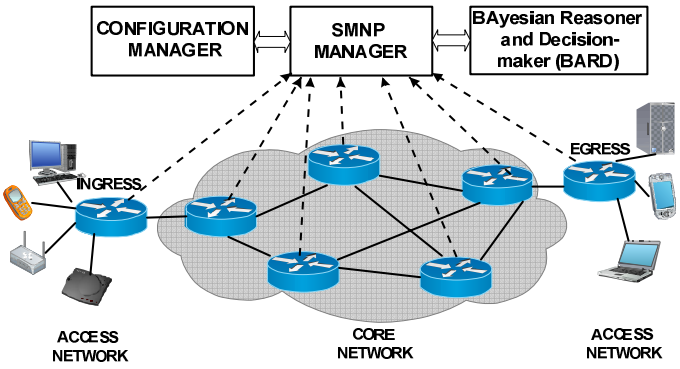

Fig. 3. Conceptual Architecture.

The collected data is essentially a time series of SNMP MIB variables defined at the various layers of the network model (e.g, Interface, IP, UDP, TCP, etc.). This monitored data provides status regarding the flow control, link utilization, QoS levels, routing protocols, etc. As discussed in section II, this data must be fed into a machine learning system to extract intelligent patterns which can assist the manager to take suitable actions. In our case, we will be using the BARD module (which implements the Bayesian learning technique) to achieve this task. The BARD module uses the structural learning feature to construct the BBN structure, learns the parameters using the parameter learning algorithm and then uses the ID for making decisions to achieve specific objectives. With time the BARD module adjusts the model parameters to reflect the changes in the real system through the sequential learning process. Two strong aspects of BARD are the predictive reasoning and diagnostic reasoning as discussed earlier. The inference results are communicated back to the SNMP manager, which in turn instructs the Configuration Manager to adjust the configuration settings of the network elements. This setup is aimed at monitoring and stabilizing the system, in case there are any specific performance deteriorations.

V. BARD Implementation

A. *Motivation*

One of the main promises of NGN is the provisioning of guaranteed QoS. It is well known that, even extremely well designed networks can suffer from performance degradation (reduced QoS) due to the congestion problem. Call Admission Control (CAC) is a preventative approach to deal with congestion, which makes decisions to accept a new call, based on whether this new call can be supported with the desired QoS [18]. This led us to use our proposed solution (BARD) for implementing a machine learning based CAC scheme.

B. *Opnet Setup*

Opnet Modeler [19] was used to simulate the network topology as shown in Fig. 4. The choice of the topology was based on the differentiated services network used in [12]. The queue length for all the routers was set to 80,000 bits and the

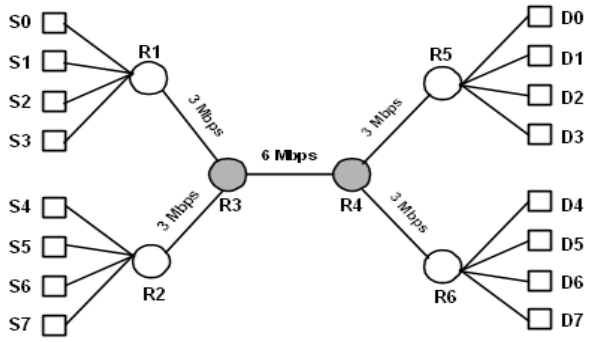

Fig. 4. Network topology simulated in Opnet.

service rate of the router queue (employing FIFO operation) was set as 9,600 bits per second.

C. Traffic Characteristics

In our simulation, the traffic characteristics of each of the sources (S0-S7) were modeled as shown in Table I. According to the settings defined for each of the sources, they were behaving as bursty sources with exponential distributions for all the duration of the simulation. This bursty nature of the traffic setup can be imagined as being similar to the traffic expected in a real NGN.

TABLE I
Characteristics of Traffic Sources (S0-S7)

Parameter	Value
Traffic Start Time (s)	Constant (5.0)
Traffic ON state time (s)	Exponential (10.0)
Traffic OFF state time (s)	Exponential (90.0)
Packet Interarrival Time (s)	Exponential (1.0)
Packet Size (bytes)	Exponential (1024)
Traffic Stop Time (s)	Never

D. Experimental Details

We ran the simulation lasting 5 hours, which was assumed to be sufficient for collecting statistics to train our BARD module. As discussed earlier in section III, our objective was to collect network statistics (through the SNMP manager) and then feed it into the BARD module, which used the learning algorithms to build the BBN. In our simulated environment (Opnet Modeler), we collected various statistics as shown in Table II, at regular intervals (approximately 20s). The BARD module then learned the BBN model based on this observed data. Even though in a real network there are many MIB variables (probably hundreds) which are monitored and collected, this experimental setup presents the general approach which can easily be extended to more variables. Another reason to select these statistics was their importance to the network manager in assessing the network performance.

VI. Simulation Results

We present the results of the simulation from two domains. One domain is the network domain which is modelled in Opnet and the other domain is the machine learning domain which is simulated in Hugin [20].

TABLE II
Network statistics which form the BBN nodes

Node Name	States	Details
Traffic_In	{low, med, high}	Aggregate traffic sent by all the traffic sources (bits/sec).
Queue_Size	{low, med, high}	The number of bits currently waiting in the queue (bits).
Packet_Loss	{low, med, high}	The number of packets rejected by the queue due to lack of capacity.
Delay	{low, med, high}	Instantaneous value of packet waiting times in the queue (s).
Traffic_Out	{low, med, high}	Traffic received over the link by the traffic sink or next router (bits/sec).

A. Data Collection and Pre-processing

The time plot of aggregate traffic (*Traffic_In*) from all the sources is shown in Fig. 5. It shows the discretized levels of *low*, *medium* and *high* which have been used to model this variable in the BBN. Similarly other network statistics shown in Table II were measured and discretized to be modelled as other BBN nodes.

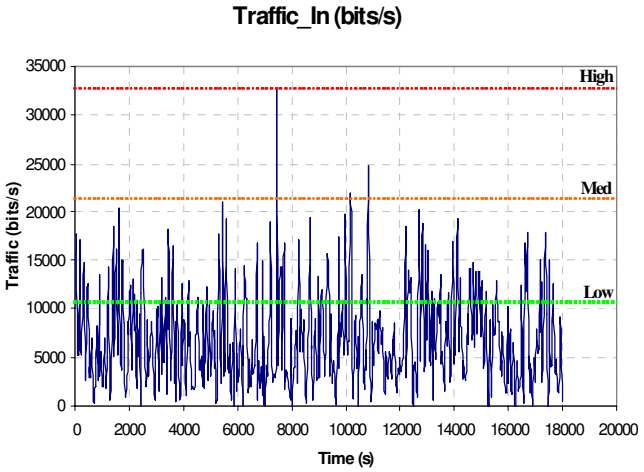

Fig. 5. Incoming Traffic, *Traffic_In* (bits/s).

B. BBN Model – Structural dependencies

The collected network data acts as input to the BARD module, which computes the BBN structure by using the NPC algorithm. The resulting BBN is shown in Fig. 6. We have considered the statistics at router R3 since it is part of the bottleneck link. The BBN structure presents the unknown dependencies between the measured variables, which cannot be normally derived by a human operator by inspection. In situations where there are many variables, this structural relationship becomes a useful tool for the network manager to study the cause and effect variables which helps him to understand the network behaviour.

C. BBN Model - The CPTs

CPTs for each of the nodes were calculated using the EM algorithm and are shown in Table III. As mentioned earlier, they provide the strength of the relationships between the

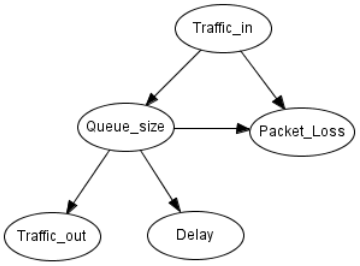

Fig. 6. BBN for router R3.

states of all the nodes of the BBN. By studying these probabilities the network manager can deduce how the various statistics affect each other. For example from Table III, it can be seen that probability that *Queue_Size* is *high*, given that *Traffic_In* is *low* is 0.057 i.e. $P(Queue\_Size = High \mid Traffic\_In = low) = 0.057$. This implies that the chances of queue size being high when the incoming traffic is low will be very low (i.e. 0.057).

TABLE III
CPTs for the nodes of the BBN (L=Low; M=Med; H=High)

Traffic_In	
Low	0.868
Med	0.127
High	0.005

Queue_Size			
Traffic_In	*Low*	*Med*	*High*
Low	0.717	0	0
Med	0.226	0.244	0
High	0.057	0.756	1

Traffic_Out			
Queue_Size	*Low*	*Med*	*High*
Low	0.577	0.004	0
Med	0.375	0.316	0
High	0.048	0.680	1

Delay			
Queue_Size	*Low*	*Med*	*High*
Low	0.741	0.066	0
Med	0.207	0.745	0.220
High	0.052	0.189	0.730

Packet_Loss									
Queue_Size	*Low*			*Med*			*High*		
Traffic_In	*L*	*M*	*H*	*L*	*M*	*H*	*L*	*M*	*H*
Low	1	0.3	0.3	1	0.9	0.3	1	0.7	0
Med	0	0.3	0.3	0	0.1	0.3	0	0.3	0.4
High	0	0.4	0.4	0	0	0.4	0	0	0.6

D. Decision-making: The Influence Diagram

We desire to use this BBN for making CAC decisions and hence we upgraded it to an ID by adding a decision node *Admission_Control* with two actions *Admit* and *Block* and a utility node named *Reward* with the utility table as shown in Table IV. Here we have chosen the *delay* variable to be involved in the decision making process. We could have even chosen the *Traffic_Out* node, which represents the link throughput. A certain action in a specific situation calls for either a reward or a penalty which are quantified accordingly. For example in Table IV, if the ID makes a decision of admitting a new call or traffic into the network when the *delay* is *low* then it gets a *reward* of 100 and likewise if it admits traffic when the *delay* is *high* then it gets a penalty of -100. The values assumed in the table are purely a matter of the system modelers' choice of how much weightage he gives to each of the action-state pairs. This reward and penalty process can be imagined to be a feedback mechanism to maximise the decision accuracy.

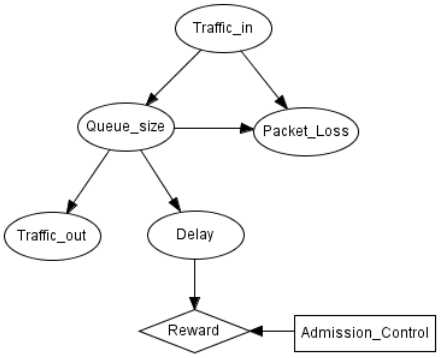

Fig. 7. Influence diagram for Call Admission Control.

TABLE IV
Utility table for *Reward*

Reward						
Admission_Control	*Admit*			*Block*		
Delay	*Low*	*Med*	*High*	*Low*	*Med*	*High*
Value	100	50	-100	-100	50	100

E. Predictive Reasoning

Now, let us see a sample decision (based on predictive reasoning) made based on the evidence of *Traffic_In* node having a *high* value. This means that the router is under an influence of heavy input traffic. Now, the ID decides not to admit any further calls into the network (with a reward of +88.42 in deciding to *Block* a call), as shown in the BBN monitor window of Fig. 8.

F. Diagnostic Reasoning

Likewise, a decision made by utilising diagnostic reasoning is demonstrated in Fig. 9, where we have the evidence that *Traffic_Out* is *low*. As can be seen the ID is now admitting a call (with a reward of +79.11) and also diagnoses the causes which are causing low throughput (as seen by the marginal probabilities in the monitor windows).

G. Discussion on results

It has been demonstrated that BBN can be used as a powerful predictive and diagnostic reasoning tool. The results provided above are the basis of corrective actions which need to be initiated by the network manager or an automated system to attain the desired goal of network performance management. As far as this paper is concerned, the scope is limited to the realization of the BARD module, by taking the data from the SNMP manager (modeled in Opnet simulation). However, this system can be easily extended to accommodate the Configuration manager (shown in Fig. 3) which can use the decisions provided by the BARD module to make appropriate

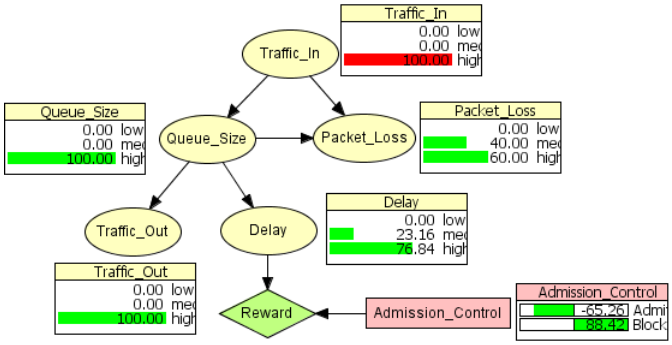

Fig. 8. A sample decision to block a call when *Traffic_In* is *high.*

configuration changes to the network elements.

H. Speed, Complexity and Scalability issues of BBN

It is to be noted that the benefits of probabilistic reasoning offered by the BBN approach comes at the following costs.

(a) Training time – Training the BBN model with historic data has to be performed initially and then occasionally (as a function of network dynamics). This training time is in the order of few 100 seconds (1.7 GHz processor) for BBN of 200 nodes of 5 states each [22].

(b) Inference time – Time required to perform Bayesian inference on observing evidences. This time is in the order of few seconds (2 GHz processor) for 200 nodes of 5 states [23].

It can be inferred that BBNs do offer intelligence at a minimal cost of processing time to train (occasionally) and infer (frequently). To this end, a detailed case study is planned for verifying the cost-benefit analysis of BBNs.

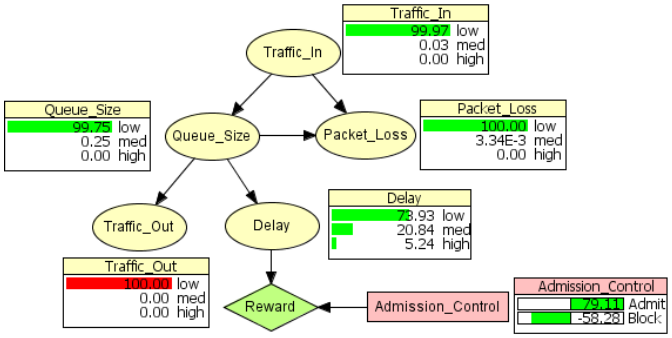

Fig. 9. A sample decision to find the probable cause of *low Traffic_Out.*

VII. CONCLUSION

This paper demonstrated that BBN are capable of representing highly dynamic systems, by efficiently modeling unknown and complex relationships between network elements. BBN can be used for predictive and diagnostic reasoning and we have shown with examples that they can facilitate decision making. As a possible extension to this work, the BBN can be populated with more network performance metrics or it can be extended to model the dependencies among multiple routers in a centralized setup. A cost-benefit analysis of BBN framework is planned in terms of algorithm processing time, CPU utilization and memory requirements. Further, we would develop BARD as a complete online learning and decision support system by integrating it with the configuration manager.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of the University of Ulster for funding this research work through the Vice Chancellors Research Studentship (VCRS).

REFERENCES

[1] *General overview of NGN,* ITU-T Recommendation Y.2001, Dec 2004.
[2] Aiko Pras et al, "Key research challenges in network management," *IEEE Communications Magazine*, Oct 2007, pp. 104-110.
[3] J. Case, M. Fedor, M. Schoffstall, J. Davin, "Simple Network Management Protocol," *RFC 1157*, IETF, May 1990.
[4] *Open systems interconnection (OSI) Common Management Information Protocol: specification,* ITU-T Recommendation X.711, Oct 1997.
[5] *TMN Management Functions,* ITU-T Recommendation M.3400, April 1997.
[6] P.G. Kulkarni, S. I. McClean, G. P. Parr, M. M. Black, "Deploying MIB data mining for proactive network management," *Proc 3rd Intl. IEEE Conference on Intelligent Systems*, Sep 2006, pp. 506-511.
[7] S. Sohail, A. Khanum, "Simplifying network management with fuzzy logic," *IEEE Itnl. Conf. on Communications*, May 2008, pp. 195-201.
[8] C. S. Hood, C. Ji , "Proactive network fault detection," *IEEE Transactions on Reliability*, Sep 1997, pp. 333-341.
[9] J. Ding, B. Kramer et al, "Predictive fault management in the dynamic environment of IP network," in *Proc IEEE International Workshop on IP Operations and Management*, Oct 2004, pp. 233-239.
[10] E. U. Ekaette, B. H. Far, "A framework for distributed fault management using intelligent software agents," *Proc IEEE Canadian Conference on Electrical and Computer Engineering 2003*, Vol 2, May 2003, pp. 797-800.
[11] P.G. Kulkarni, S.I. McClean, G.P. Parr, M.M. Black, "Proactive predictive queue management for improved QoS in IP networks," *Proc of IEEE ICN/ICONS/MCL*, Apr 2006, pp. 2-7.
[12] T. C.-K. Hui, Chen-Khong Thanm , "Adaptive provisioning of differentiated services networks based on reinforcement learning ," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 33, no. 4, Nov 2003, pp. 492-501.
[13] P. Spirtes, C. Glymour, R. Scheines, *Causation, Prediction, and Search*, MIT Press, Second edition, Jan 2001.
[14] Steck, H., "Constrained-based structural learning in Bayesian networks using finite data sets", PhD Thesis, 2001, Institut für der Informatik der Technischen Universität München.
[15] S. L. Lauritzen, "The EM algorithm for graphical association models with missing data," *Computational Statistics & Data Analysis*, Vol. 19, 1995, pp. 191-201.
[16] D. J. Spiegelhalter & S. L. Lauritzen, "Sequential updating of conditional probabilities on directed graphical structures," *Networks, Special Issue on Influence Diagrams*, Vol. 20, No. 5, Aug 1990, pp. 579-605.
[17] J. Pearl, *Probabilistic reasoning in intelligent systems,* Morgan Kaufmann, San Mateo, CA, 1988.
[18] H. G. Perros, K. M. Elsayed, "Call admission control schemes," *IEEE Communications Magazine*, Nov 1996, pp. 82-91.
[19] Available at: http://www.opnet.com
[20] Available at: http://www.hugin.com
[21] S. L. Lauritzen, D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society,* Series *B (Methodological)*, Vol. 50, No. 2, Jan 1988, pp. 157-224.
[22] A.L. Madsen et al, "The Hugin tool for learning Bayesian networks," *LNCS*, Vol 2711, Springer 2004, pp. 594-605.
[23] S. F. Galan, "Belief updating in Bayesian networks by using a criterion on minimum time," *Pattern Recognition Letters*, Vol 29, 2008, pp. 465-482.
[24] D. Heckerman, "A tutorial on learning with Bayesian networks," *Learning in Graphical Models*, M. Jordan, ed. MIT Press, Cambridge, MA, 1999.