This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Cognitive Network Inference through Bayesian Network Analysis

Giorgio Quer[§⋆], **Hemanth Meenakshisundaram**[⋆], **Bheemarjuna Tamma**[⋆], **B. S. Manoj**[⋆], **Ramesh Rao**[⋆], **Michele Zorzi**[§⋆]
[§]DEI, University of Padova, via Gradenigo 6/B – 35131, Padova, Italy.
[⋆]University of California at San Diego, La Jolla, CA 92093.

***Abstract*—Cognitive networking deals with applying cognition to the entire network protocol stack for achieving stack-wide as well as network-wide performance goals, unlike cognitive radios that apply cognition only at the physical layer. Designing a cognitive network is challenging since learning the relationship between network protocol parameters in an automated fashion is very complex. We propose to use Bayesian Network (BN) models for creating a representation of the dependence relationships among network protocol parameters. BN is a unique tool for modeling the network protocol stack as it not only learns the probabilistic dependence of network protocol parameters but also provides an opportunity to tune some of the cognitive network parameters to achieve desired performance. To the best of our knowledge, this is the first work to explore the use of BNs for cognitive networks. Creating a BN model for network parameters involves the following steps: sampling the network protocol parameters (Observe), learning the structure of the BN and its parameters from the data (Learn), using a BN-based inference engine (Plan and Decide) to make decisions, and finally effecting the decisions (Act). We have proved the feasibility of achieving a BN-based cognitive network system using the ns-3 simulation platform. From the early results obtained from our approach, we provide interesting insights on predicting the network behavior, including the performance of the TCP throughput inference engine based on other observed parameters.**

I. Introduction

Cognitive networking [1], [2] deals with wireless systems that will have deeper awareness of their own operations and of the network environment, learn relationships among network parameters, plan and make decisions in order to achieve local, end-to-end, and network-wide performance as well as resource management goals. In our concept of cognitive network, all networking elements track the spatial, temporal, and spectral dynamics of their own behavior and the behavior associated with the environment. The information so gathered is used to learn, plan and act in a way that meets network or application requirements. Cognitive networking differs from cognitive radios or cognitive radio networking in that the latter two typically apply cognition only at the physical layer to dynamically detect and use spectrum holes, focusing strictly on dynamic spectrum access, whereas the objective of cognitive networks is to apply cognition to the entire network protocol stack for achieving network-wide performance goals.

Cognition in the traditional protocol stack is challenging. First, the probabilistic relationships among the various parameters that span across the entire protocol stack are not clearly understood. Second, the tools that can be used to determine such complex relationships are not well known. The traditional layered protocol stack has helped establish an order and structure in the role of various protocols and hence has contributed greatly to the faster progress of networking. In this work, we propose an architecture for cognitive networking that can be integrated with the existing layered protocol stack and that exploits new and hitherto unused tools from artificial intelligence. Specifically, we suggest the use of a probabilistic graphical model for modeling the layered protocol stack. The use of graphical models was mentioned in [3], however the main focus of that paper is on the exploitation of a Monte Carlo method, Simulated Annealing, for increasing the performance of the protocol stack. In this paper instead we use Bayesian Network (BN), a graphical representation of statistical relationships between random variables, widely used for statistical inference and machine learning [4]. A graphical model is represented by a graph structure consisting of vertices and edges, and conditional probability distributions. The structure of the model consists of the specification of a set of conditional independence relations for the probability model, represented by a set of missing edges in the graph. If a variable x_i does not depend directly on variable x_j, then there is no edge between them. Conditional probabilities are used to capture the statistical dependence between variables. The joint probability distribution of a set of random variables can be easily obtained using the chain rule.

The use of BN for modeling the protocol stack provides us with a unique tool to learn not only the influence of certain parameters on others, but also to apply the inferred knowledge to achieve the desired level of performance at the higher layers. For example, our modeling enables a node to determine which combinations of lower layer parameters are useful for achieving a certain higher layer throughput performance.

The main contributions of this paper are:

1. the integration of BN into our Cognitive Network framework;
2. the application of BN to study network parameters in a realistic wireless LAN scenario;
3. a performance analysis of the BN inference engine's accuracy in such a scenario.

In this paper we focus on a single-hop network, while the multi-hop case is studied in [5]. The rest of this paper is organized as follows: Section II summarizes the fundamentals of BN models, Section III discusses our architecture for cognitive networking in detail, Section IV presents the application of BN learning to a WLAN scenario and Section V shows the

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

performance of the inference engine to predict TCP throughput. We conclude the paper in Section VI.

II. Bayesian Network preliminaries

In this section we summarize some techniques from machine learning and Bayesian analysis [4] that will be used to design the dependence structure of the underlying relationships that probabilistically connect a set of random variables. In particular, our aim is to design a Bayesian Network (BN), a graphical model for representing conditional independences between the variables through a Directed Acyclic Graph (DAG). This graph will be used to efficiently compute marginal and conditional probabilities that are required for inference. A node in the DAG represents a random variable, while an arrow that connects two nodes represents a direct probabilistic relation between the two corresponding variables. Node i, representing the random variable x_i, is a parent of node h if there exists a direct arc from i to h and we write $i \in \text{pa}_h$, where pa_h is the set of parents of node h. From the graph it is always possible to determine the conditional independence between two variables, applying a set of rules known as *d-separation* rules, e.g., see [4], [6] for a detailed description about BNs properties.

In this section we assume to have M discrete variables, $x_1, \ldots, x_M$, with unknown dependence relations, and we write the realization of the variable x_i at time k as $x_i^{(k)}$. We assume to deal with a complete dataset, i.e., the collection of M column vectors of length N, where N is the number of the independent realizations of the variables at time samples that for simplicity we indicate as $k = 1, \ldots, N$. In other words we know the values of all the elements $x_i^{(k)} \in \mathcal{D}_{N,M}$, with $i \in \{1, \ldots, M\}$ and $k \in \{1, \ldots, N\}$. The first step for designing the BN is to learn from the dataset the *qualitative* relations between the variables and their conditional independences, in order to represent them in a DAG. The second step, given the DAG, is to infer the *quantitative* relations in order to obtain a complete probabilistic structure that describes the variables of interest. With this structure we can calculate all the joint and conditional probabilities between the variables of the network, as explained in Section II-C.

A. Structure Learning

A BN represents graphically in a DAG the mutual conditional independence relations of a set of variables. In the literature there are two methods to design such DAG through a learning process known as structure learning. The former is the constraint based method [6], in which a set of conditional independence statements is established based on some a priori knowledge or on some calculation from the data. Then this set of statements is used to design the DAG following the rules of *d-separation*. An alternative method, commonly used in the absence of a set of given conditional independence statements, is the score based method [7], that is able to infer a suboptimal DAG from a sufficiently large data set $\mathcal{D}_{N,M}$. This method consists of two parts:

1) a function that scores each DAG based on how accurately it represents the probabilistic relations between the variables based on the realizations in the dataset $\mathcal{D}_{N,M}$;
2) a search procedure, that selects which DAGs to score within the set of all possible DAGs.

The score function should be computationally tractable and should balance the accuracy and the complexity of the structure, i.e., the number of arrows in the graph. It is worth to highlight that a complete DAG, with all nodes connected, can describe all possible probabilistic relations among the nodes and that it is the absence of an arrow that brings information in the form of conditional independence. In this paper we have chosen the Bayesian Information Criterion (BIC) as a score function, that is easy to compute, is based on the Maximum Likelihood (ML) criterion, i.e., how well the data suits a given structure, and penalizes DAGs with a higher number of edges. In general it is defined as [8]:

$$\text{BIC}(S|\mathcal{D}_{N,M}) = \log_2 P(\mathcal{D}_{N,M}|S, \hat{\theta}_S) - \frac{\text{size}(S)}{2} \log_2(N) \ , \tag{1}$$

where S is the DAG to be scored, $\text{size}(S)$ is the number of edges in S, $\mathcal{D}_{N,M}$ is the dataset, $\hat{\theta}_S$ is the ML estimation of the parameters of S and N is the number of realizations for each variable in the dataset. In the case in which all the variables are multinomial, with a finite set of outcomes r_i for each variable x_i, we define q_i as the number of configurations over the parents of x_i in S, i.e., the number of different combinations of outcomes for the parents of x_i. We define also N_{ijk} as the number of outcomes of type k in the dataset $\mathcal{D}_{N,M}$ for the variable x_i, with parent configuration of type j and N_{ij} as the total number of realizations of variable x_i in $\mathcal{D}_{N,M}$ with parent configuration j. Given these definitions, it is possible to rewrite the BIC for multinomial variables as [7]:

$$\begin{aligned} &\text{BIC}(S|\mathcal{D}_{N,M}) \\ &= \sum_{i=1}^{M} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log_2 \left(\frac{N_{ijk}}{N_{ij}} \right) - \frac{\log_2 N}{2} \sum_{i=1}^{M} q_i (r_i - 1) \ , \end{aligned} \tag{2}$$

which is a computationally tractable function that reduces the scoring function to a counting problem. A detailed comparison of scoring functions for structure learning can be found in [9].

Unfortunately, it is not possible to score all the possible DAGs given a set of M random variables, since an exhaustive enumeration of all structures is not possible for large values of M, as the total number of possible DAGs, $f(M)$, increases super exponentially with M:

$$f(M) = \sum_{i=1}^{M} (-1)^{i+1} \frac{M!}{(M-i)!i!} 2^{i(M-i)} f(M-i) \ . \tag{3}$$

For this reason, it is necessary to define a search procedure that selects a small representative subset of the space of all DAGs. There exist plenty of search strategies based on heuristics consisting of different methods to explore the search space and look for a local maximum of the score function, but in general these heuristics do not provide any guarantee of finding a global maximum for the score function on the space of possible DAGs, as highlighted in [10]. In this paper we use a simple heuristic as detailed in Section IV.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

B. Parameter Learning

Parameter learning is a phase of the learning procedure that consists in estimating the best set of parameters for each variable, given the independence relations defined by the DAG. According to the definition of BN, each variable is directly conditioned only by its parents, so the estimation of the parameters for each variable x_i should be performed only conditioned on the set of its parents pa_i in the chosen DAG. There are two main methods, both asymptotically equivalent, consistent and suitable to be implemented in an on-line manner, given a sufficient size of the dataset $\mathcal{D}_{N,M}$, namely ML and Bayesian estimation given Dirichlet priors [10]. In this paper we choose ML for parameter learning, coherently with the choice of BIC as a scoring function for the structure learning algorithm. In the case of multinomial variables we can write with an abuse of notation[1] the ML estimation as:

$$\hat{\theta}_{x_i=k|\text{pa}_i=j} = \frac{N_{ijk}}{N_{ij}} \simeq P\left[x_i = k|\text{pa}_i = j\right] \ . \tag{4}$$

C. Inference with Bayesian Networks (BNs)

After performing the structure learning and parameter learning phases on the dataset $\mathcal{D}_{N,M}$, we obtain a complete BN that represents the probabilistic relation between the variables x_i with $i \in \{1, \ldots, M\}$. At this point we can use the BN to compute marginal and conditional probabilities on the variables x_i in order to make inference. For example, given a singly connected[2] linear BN with $M = 4$, with connections $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, we can apply *belief propagation* [11] for inference, according to the d-separation rules. We observe at time k the evidence $x_1 = x_1^{(k)}$ and $x_2 = x_2^{(k)}$ and we want to infer the realization of x_4. The belief propagation algorithm updates the marginal probabilities of all the variables in the network based on the given evidence, yielding as output the probabilities:

$$P\left[x_4 = x_4^{(k)}|x_1, x_2\right] = P\left[x_4 = x_4^{(k)}|x_2\right] \ . \tag{5}$$

These probabilities will be used by the inference engine detailed in Section IV-C to make predictions for some network parameters in the Cognitive Network framework.

III. Cognitive Network Architecture

Here we present the network architecture in which we want to integrate the BN tools. The network parameters within a stack or within a particular protocol can be classified into two basic categories: observable parameters and controllable parameters. The observable parameters provide important information about the characteristics or behavior of the protocol and the status of the network system. For example, in the MAC layer, the average packet retransmission count provides information about the packet losses and retransmissions; however, it cannot be directly controlled, although we can set the maximum retransmission limit. Examples of controllable parameters include the TCP Congestion Window (T.CW), with its minimum and maximum levels allowed. Note that the controllable parameters in a given layer may affect the observable parameters in some other layer.

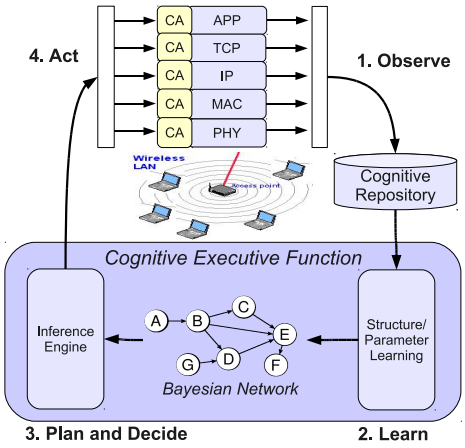

Fig. 1. The Cognitive Network Architecture.

Our approach uses a fully distributed solution where software modules called Cognitive Agents (CA) are plugged into each layer as well as protocols of importance, so that we have access to the protocol parameters in each layer. Communication among the CAs is coordinated through a backplane called CogPlane [2]. Each CA is responsible for periodically sampling a set of protocol parameters and repositorizing the sampled data to a *Cognitive Repository*. The CogPlane contains a Cognitive Execution Function (CEF), that serves as the brain of the cognitive network protocol stack and executes the optimization of protocols within the stack. We decompose the cognitive action into the four phases of the "Cognition Cycle" [12]: 1) Observe, 2) Learn, 3) Plan and Decide, and 4) Act.

Fig. 1 shows the architecture of the cognitive network protocol stack that (in the Observe phase) collects parametric information from each layer and (in the Learn phase) learns the effect of controllable parameters on observable parameters and (in the Plan and Decide phase) determines the values to be assigned to various controllable parameters in order to meet the performance requirements imposed by the application layer, and finally (in the Act phase) reconfigures the network elements.

The Observe phase includes in-stack and out-of-stack parameter observation. The observe action, within the protocol stack, is done by periodically sampling the observable and controllable protocol parameters across all layers. For in-stack parameter observation, the CAs will be designed to periodically sample the state of the network parameters. The main out-of-stack parameter is the wireless traffic information which is collected by the CA in the physical layer.

The Learn phase consists of the process of building the BN structural relationship between the observable and controllable network parameters within the network protocols, network layers, and environment parameters such as spectrum, location, and time. Our approach is to first identify a set of important parameters from each layer of the protocol stack or from representative protocols in each layer. Once the temporal behavior data of such parameters is collected in a given spatial

[1] With the notation $\text{pa}_i = j$ we mean that the parents of node i are in the configuration of type j.

[2] A singly connected network is a network in which the undirected graph has no loops.

domain, the challenge is to derive critical causality relationships between parameters. We use the above discussed BN model for deriving the critical causality relationship between parameters. The BN learning model for some in-stack network parameters is discussed in Section IV.

The Plan and Decide phase of the system focuses on decomposing the network or user performance objectives into real actionable information within the controllable parameter set of the entire protocol stack. The spatio-temporal-spectral characteristics derived from historical information, in the form of the probabilistic structure in the graphical models, will provide significant knowledge for this phase. The Plan and Decide phase would translate abstract objectives, e.g., minimum throughput required or maximum end-to-end delay that can be tolerated by the application layer protocol, into the real controllable network parameters, e.g., combinations of multi-layer protocol parameter values. A key role in this phase is the prediction of the values of the network parameters of interest given the observations, in order to choose the appropriate actions to perform in the network. The inference engine, responsible for this prediction, is presented in Section IV and its performance is tested in SectionV.

Finally, in the Act phase the decision is effected. Compared to the other phases, this phase is far less complex. For example, in this phase, the CEF sends instructions to be executed at all or selected layers or protocols in the form of identified controllable parameters and their suggested values. As an example, the initial congestion window and slow start threshold of TCP can be instructed to a new node based on the network environment's spatio-temporal characteristics [2]. The process of Act may be carried out at different scales on different devices. For example, a client node in a Wireless LAN uses this framework to observe local node parameters and to optimize certain flows of that node, whereas a wireless AP can use LAN wide parameters to help optimize the network for certain high priority client nodes.

IV. Learning a Bayesian model for Cognitive Networks

In this section, we look at how the BN structure and parameter learning algorithms are used in the learning phase to infer the probabilistic relations between the MAC and TCP parameters of the network. While we focus on the relationship between TCP and MAC as an example in this paper, our strategy can be generalized for the entire protocol stack.

A. Network Scenario

We analyze a scenario with up to 40 active users that produce 20 independent TCP flows within the WLAN and communicate through one Cognitive Access Point (CogAP). The CogAP is responsible for learning a suitable Bayesian Network (BN) based on the data collected at each active node and for taking consequent decisions to optimize the traffic in the network. The 20 TCP flows are simulated using the ns-3 discrete event network simulator [13], augmented with hooks that collect the values of selected TCP and MAC parameters at regular sampling intervals for each flow. In particular, the TCP parameters collected are the following: x_1 is the congestion window value (T.CW), i.e., the total amount of data [bytes] that can remain unacknowledged at any time; x_2 is the congestion window status (T.CWS), which can take one of three values based on the TCP algorithm: linear increasing, exponentially increasing or exponentially decreasing; x_3 is the Round Trip Time (T.RTT), i.e., the last value registered in the sampling interval of the time between when a packet is sent and when the corresponding acknowledgement is received; x_4 is the throughput (T.THP), i.e., the total amount of unique data [bytes] acknowledged in a sampling interval. The selected MAC parameters are the following: x_5 is the Contention Window (M.CTW), i.e., the maximum number of slots the node will wait before transmitting a packet at the MAC level, according to a random back off interval between zero and CW; x_6 is the number of MAC transmissions (M.TX), i.e., the total number of original MAC packets transmitted in the sampling interval; x_7 is the number of MAC retransmissions (M.RTX), i.e., the total number of MAC retransmissions in the sampling interval. Among the parameters described above, T.CWS, T.THP, M.CTW, M.TX, and M.RTX have a finite number of outcomes, less than or equal to $n_q = 30$. Also the remaining two parameters, T.CW and T.RTT, have been quantized to n_q levels, in order to apply the structure and parameter learning algorithms for multinomial variables, since they are computationally more efficient and require less training data. These parameters have been quantized in n_q levels chosen according to their estimated n_q-quantiles[3]. We perform the learning methods in two datasets, namely $\mathcal{D}^1_{N,M}$ and $\mathcal{D}^2_{N,M}$, that differ by the sampling intervals in which we register the network parameters values, $\Delta T_1 = 100\ ms$ and $\Delta T_2 = 1000\ ms$, respectively. The number of samples is of the order of $N \simeq 5 \times 10^4$ samples and the number of parameters is $M = 7$ in both datasets.

B. Designing the Bayesian Network from the data

Structure learning is performed by a score based method implemented in Matlab and available in [15]. This phase is the most computationally demanding, as for $M = 7$ parameters it requires to discriminate from a set of about 1.1×10^9 DAGs. We have used three different search procedures, namely 1) Hill Climbing (HC), 2) a Markov Chain Monte Carlo method (MCMC) and 3) a simple heuristic. The first two are classical methods described in [16] and implemented in [15]. The simple heuristic we propose in this paper consists in dividing the net into two subnets, one for the MAC parameters ($M_{MAC} = 3$) and one for the TCP parameters ($M_{TCP} = 4$), and finding the best structure for each one of them, separately scoring all the possible DAGs, that for each subnet are less than 10^3. Then these two separate subnets form the initial structure given as input to the HC algorithm. Each search procedure gives a DAG as a result and we choose among these three DAGs the one with the highest BIC score in Eq. (2). From dataset $\mathcal{D}^1_{N,M}$ we obtained the DAG depicted in Fig. 2-(a) that represents the static connection between the parameters sampled at intervals

[3]Since it is unrealistic to introduce a complex non-uniform quantizer in each node, we have chosen this quantizer that performs better than the uniform one in the case of non uniform realistic variables, as suggested by intuition looking at the cdf of the variables, and as formally explained in [14].

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

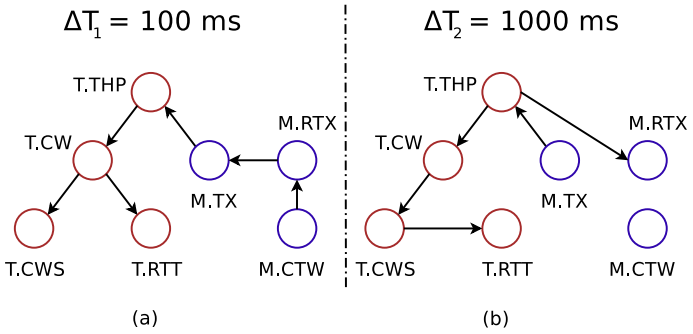

Fig. 2. BNs learned (a) from the dataset $\mathcal{D}^1_{N,M}$ with sampling time $\Delta T_1 = 100\ ms$ and (b) from the dataset $\mathcal{D}^2_{N,M}$ with $\Delta T_2 = 1000\ ms$.

of $\Delta T_1 = 100\ ms$. From dataset $\mathcal{D}^2_{N,M}$ instead we obtain the DAG depicted in Fig. 2-(b), that is slightly different from the previous one. The reasons for this difference are that when a longer sampling period is used, the data shows that there is a direct probabilistic relation between the TCP throughput (T.THP) and the number of MAC retransmissions (M.RTX), while the MAC contention window value (M.CTW) becomes independent of the other parameters. In the graph the former translates into the appearance of a direct edge between M.RTX and T.THP and the latter translates into the disappearance of the edge connecting M.CTW with M.RTX. All the other differences in the graph can be explained similarly.

C. Learning the Inference Engine

The Plan and Decide phase of our cognitive network is implemented using a Bayesian inference engine, that is able to predict the value of certain network parameters based on the observation of some other parameters. The inference engine is designed using the parameter learning algorithm described in Eq. (4), that defines quantitatively the probability relations among the parameters based on the given training set. The estimate of an unobserved parameter x_i is the expectation of such parameter based on the probability mass function (pmf) learned using Eq. (4), where the probability is conditioned on the set of observed parameters, i.e., the evidence X_e. The estimated value for x_i at time k becomes:

$$\hat{x}_i^{(k)} = E\left[x_i^{(k)} | X_e\right] \ . \tag{6}$$

V. Performance Evaluation of the Inference Engine

The Bayesian inference engine exploits the BN structure to infer the expected values of certain important parameters based on the evidence. We apply the inference engine to the two different DAGs that connect the network parameters in the cases of sampling interval $\Delta T_1 = 100\ ms$ and $\Delta T_2 = 1000\ ms$, depicted in Fig. 2. Now we use such structures to infer the value of the TCP throughput (T.THP) based on some evidence, i.e., the measured values of a subset, possibly empty, of the network parameters, X_e. In Fig. 3 we show the measured and the estimated T.THP for 40 consecutive samples, with sampling

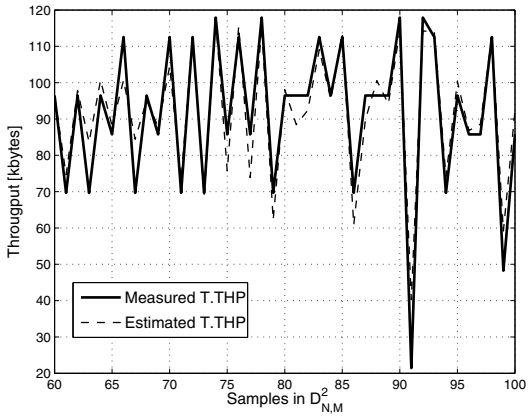

Fig. 3. Measured value and estimated value of T.THP (with evidence $X_e = \{\text{all}\}$) for 40 consecutive samples in $\mathcal{D}^2_{N,M}$.

time ΔT_2, in the case where we have as evidence all the other network parameters. In order to measure the accuracy of the T.THP estimate, we define the normalized Root Mean Square Error (RMSE), i.e.:

$$\xi = \frac{\sqrt{\sum_{k=1}^{N}\left(\hat{x}_4^{(k)} - x_4^{(k)}\right)^2}}{\sqrt{\sum_{k=1}^{N}\left(x_4^{(k)}\right)^2}} \ , \tag{7}$$

where $x_4^{(k)}$ is the actual value of T.THP at time k and $\hat{x}_4^{(k)}$ is the estimated value based on the evidence in the set X_e. In the simulations we used the data collected from the ns-3 simulator mentioned in Section IV, while the inference engine is written in Matlab. In Figs. 4 and 5 we represent the performance of the inference engine for the TCP throughput estimate for $\Delta T_1 = 100\ ms$ and $\Delta T_2 = 1000\ ms$, respectively. In the x-axis we vary the length of the training set for parameter learning, in logarithmic scale, and in the y-axis we represent the normalized RMSE calculated as in Eq. (7). The total number of sample intervals used in this simulation is $N = 4{\cdot}10^4$. The curves in the graphs correspond to different evidence subsets for the estimate. In particular, we analyze the performance of the inference engine in these cases: 1) $X_e = \emptyset$, there is no evidence and the T.THP is estimated based on its marginal distribution; 2) $X_e = \{\text{T.RTT}\}$, the evidence is the TCP RTT; 3) $X_e = \{\text{T.CW}\}$, the TCP Congestion Window; 4) $X_e = \{\text{M.RTX}\}$, the number of MAC retransmissions; 5) $X_e = \{\text{M.TX}\}$, the number of MAC transmissions; 6) $X_e = \{\text{T.RTT, T.CW, T.CWS, M.TX, M.RTX, M.CTW}\}$, where all the parameters but the TPC throughput are observed. In particular in Fig. 4, with $\Delta T_1 = 100\ ms$, we have a small improvement compared to the case of no evidence when we measure the number of MAC retransmissions, while we have a bigger improvement when measuring the number of MAC transmissions, which is the parameter most related to TCP throughput, as expected in our scenario. It is interesting to notice that for a training set smaller than $2 \cdot 10^3$ samples the inference engine performs better with the knowledge of only M.TX than with the knowledge of all the parameters. In order

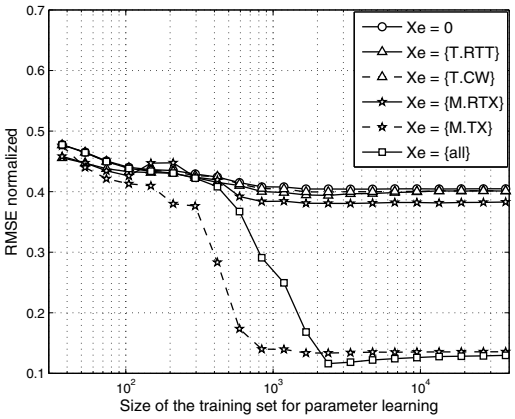

Fig. 4. Performance of the T.THP inference engine for $\Delta T_1 = 100\ ms$.

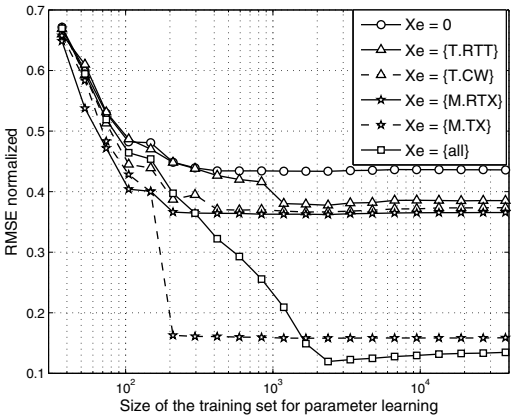

Fig. 5. Performance of the T.THP inference engine for $\Delta T_2 = 1000\ ms$.

to explain this behavior, we should notice from Fig. 2-(a) that M.TX and T.CW separate T.THP from the rest of the network, according to the rules of *d-separation*. Moreover, when the training set is too short the estimates of the values of the variables are not precise, so that including additional variables (T.CW in this case) adds potentially inaccurate information which negatively affects the overall throughput estimate performance. Instead, with $\Delta T_2 = 1000\ ms$ (see Fig. 5), the (now more accurate) knowledge of T.CW and T.RTT does provide some advantages for throughput inference, even though M.TX still has a dominant role in the considered single-hop scenario.

Observing the performance results we conclude that our inference engine helps the Plan and Decide phase not only in predicting the behavior of certain network parameters but also in guaranteeing quality of T.THP estimate with proper choice of the evidence. A key application to exploit the T.THP inference engine is to adapt the controllable parameters such as T.CW, T.CWS and M.CTW to achieve the desired T.THP.

VI. Conclusions

In this work we proposed the integration of the Bayesian Network (BN) model into an architecture for Cognitive Networking. Cognitive networks learn their operating network environment and the network protocol parameter relationships in order to achieve network-wide performance objectives. In this context, we have shown that the use of BN is very promising for learning the probabilistic structure and designing an inference engine for the chosen parameters. The inference engine can be exploited to achieve performance goals like a minimum guaranteed level for the TCP throughput. The results support the following observations: (i) BN is a useful tool for cognitive networking to determine, represent and exploit the probabilistic dependencies and conditional independencies between protocol parameters, (ii) exploiting BN, we can design an inference engine to accurately predict the behavior of protocol parameters, and (iii) we can obtain useful insights on the influence of the data size used for training on the accuracy of network parameter behavior prediction. Our future plans include the following: (a) implementing the Act phase of the cognitive network operation, (b) studying more complex BN models to also describe the time dependencies between network parameters, (c) investigating other methods for efficient inference over BN, (d) improving the prediction accuracy, and (e) finding creative ways of using the inference engine to optimize protocol parameters in a cross-layer Cognitive framework.

Acknowledgments

This work was partially supported by the U.S. Army Research Office, Grant. No. W911NF-09-1-0456 and UCSD-CWC (Center for Wireless Communications).

References

[1] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, "Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives," *IEEE Communications Magazine*, vol. 44, no. 12, pp. 51–57, December 2006.

[2] B. Manoj, R. Rao, and M. Zorzi, "Cognet: a cognitive complete knowledge network system," *IEEE Wireless Communications*, vol. 15, no. 6, pp. 81–88, December 2008.

[3] E. Meshkova, J. Riihijärvi, A. Achtzehn, and P. Mähönen, "Exploring Simulated Annealing and Graphical Models for Optimization in Cognitive Wireless Networks," in *IEEE Globecom 2009*, Nov.-Dec. 2009.

[4] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] G. Quer, H. Meenakshisundaram, B. Tamma, B. S. Manoj, R. Rao, and M. Zorzi, "Using Bayesian Networks for Cognitive Control of Multi-hop Wireless Networks," in *MILCOM 2010*, San Jose, CA, US, Nov. 2010.

[6] D. Margaritis, "Learning Bayesian Network Model Structure from Data," Ph.D. dissertation, School of Computer Science - Carnegie Mellon University, Pittsburgh, PA, May 2003.

[7] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*. Springer, 2007.

[8] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[9] S. Yang and K.-C. Chang, "Comparison of score metrics for Bayesian network learning," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* , vol. 32, pp. 419–428, May 2002.

[10] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[11] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2007.

[12] J. Mitola III, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," Ph.D. dissertation, Royal Institute of Technology (KTH), Sweden, May 2000.

[13] http://www.nsnam.org/.

[14] D. Lampasi, "An Alternative Approach to Measurement based on Quantile functions," *Measurement*, vol. 41, no. 9, pp. 994–1013, 2008.

[15] K. Murphy, "Bayes Net toolbox for Matlab," Last time accessed: July 2010. [Online]. Available: code.google.com/p/bnt/

[16] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.