Using Bayesian Networks for Cognitive Control of Multi-hop Wireless Networks

Giorgio Quer[§⋆], **Hemanth Meenakshisundaram**[⋆], **Bheemarjuna R. 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, USA

***Abstract*—Tactical communication networking faces diverse operational scenarios where network optimization is a very challenging task. Learning from the network environment, in order to optimally adapt the network settings, is an essential requirement for providing efficient communication services in such environments. Cognitive networking deals with the application of cognition to the entire protocol stack for achieving network-wide performance goals. One of the key requirements of a cognitive network is to learn the relationships between network protocol parameters spanning the entire stack in relation with the operating network environment. In this paper, we use a probabilistic graphical modeling approach, Bayesian Networks (BNs), in order to create a representation of the dependence relationships between significant parameters spanning transport and medium access control (MAC) layers in multi-hop wireless network environments. We exploit this model to face one of the problems of the TCP protocol, that does not have any mechanism to infer when congestion is about to occur in the network and therefore waits till some packets are lost for reacting to congestion in the network. Such a reactive nature of TCP leads to wastage of precious network resources like bandwidth and power. In this paper we show how to infer in advance the congestion state of the network. We constructed BNs for different network environments by sampling network parameters on-the-fly in the ns-3 simulation platform. We found that it is possible to predict the congestion state of the network with quite good accuracy given sufficient training samples and the current value of the TCP congestion window.**

I. Introduction

Cognitive networking [1], [2] is an emerging paradigm that deals with how wireless systems learn relationships among network parameters, network events, and observed network performance, plan and make decisions in order to achieve local, end-to-end, and network-wide performance as well as resource management goals. In cognitive networks, all nodes track the spatial, temporal, and spectral dynamics of their own behavior, as well as the environment. The information so gathered is used to learn, plan and act in a way that meets network or application Quality of Service (QoS) requirements.

One of the key requirements of a cognitive network is to learn the relationships among network protocol parameters spanning the entire stack in relation with the operating network environment. In this paper, we use a probabilistic graphical modeling approach, Bayesian Networks (BNs), in order to create a representation of the dependence relationships between significant network parameters in multi-hop wireless network environments. As an example, tactical communication environments are dynamic and very challenging, and there is no predefined protocol configuration that optimally operates in these conditions. In these scenarios, learning the optimal parameter set for the network protocol stack, by using the BN structure constructed from historical network behavior, can help efficiently meet QoS requirements.

Cognitive networking is different from cognitive radios or cognitive radio networking in that the latter two typically apply cognition only at the PHY layer to dynamically detect and use spectrum holes, and focus strictly on dynamic spectrum access. We notice that there are still some open problems to solve before cognition can be applied to the entire protocol stack. 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.

In this work, we propose a cognitive network node architecture that can be integrated with the existing layered protocol stack. Our work partially addresses the requirement of modeling the layered protocol stack using new and hitherto unused tools from artificial intelligence. Specifically, we consider the use of BNs, a graphical representation of statistical relationships among random variables, widely used in machine learning [3].

The use of BN for modeling the protocol stack provides us with a unique tool, not only to learn the influence of certain parameters on others, but also to apply the inferred knowledge and achieve a certain 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. In a related paper [4], we applied the BN approach to a single hop Wireless LAN (WLAN) scenario. In this paper, while using the same BN tool and cognitive architecture, we significantly improve on that work, analyzing a realistic multi-hop wireless network scenario and predicting TCP's congestion status from the current state of the network for proactively making decisions to adjust the value of the congestion window.

II. Cognitive Network Framework

In this section, we present the cognitive network architecture [4], designed with the goal of optimizing the network performance by carefully tuning the values of the controllable parameters within the network stack, through the observation of the observable parameters and the exploitation of knowledge acquired from historical network behavior. Examples of observable parameters are the number of packets transmitted at the MAC layer (M.TX[1]), or the number of retransmissions at the

[1]We define a parameter format, X.ABC, where X represents the layer of operation and ABC represents the protocol parameter in layer X. For example, the transport layer parameter Round Trip Time (RTT) is represented as T.RTT and the MAC layer contention window is represented as M.CTW.

978-1-4244-8180-4/10/$26.00 ©2010 IEEE

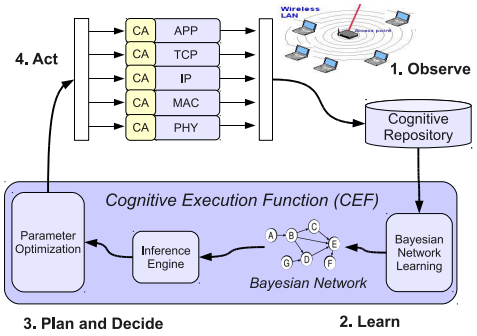

Fig. 1. Cognitive Network Node Architecture.

MAC layer (M.RTX). Controllable parameters are for example the TCP Congestion Window (T.CW), or the MAC Contention Window (M.CTW). These parameters can be observed and appropriately tuned by a specific protocol able to predict their optimal values, based on the status of the network and on the past history.

The cognitive network node architecture is shown in Fig. 1, with the functional modules that realize the four phases of the "cognition cycle" [5]: 1) Observe, 2) Learn, 3) Plan and Decide and 4) Act. In the Observe phase we adopt a fully distributed solution where software modules, the Cognitive Agents (CA), are plugged into each layer to have access to the protocol parameters of importance, that are periodically sampled and saved into a local cognitive database repository.

The Cognitive Execution Function (CEF) is the brain of the cognitive network node where the optimization decisions for protocols within the stack are made. CEF realizes the Learn and the Plan and Decide phases of the cognition cycle. The Learn phase is a key phase in the cognition process, where the cognitive node exploits the information collected in the Observe phase to infer a probabilistic structure that connects the parameters of interest, using the BN model described in Section III. In the Plan and Decide phase, the cognitive node makes use of the BN model derived from historical data for predicting future values of the parameters of interest based on the current network status. The inference engine, see Fig. 1, receives as input the observed parameters at a given discrete time k and gives as output the future value of one or more parameters of interest. This prediction can be exploited in the Plan and Decide phase to optimize a controllable parameter, or to predict an unwanted behavior of the network, e.g., congestion, and take the appropriate actions in the protocol stack before this happens. Finally, in the Act phase, the decisions made in the Plan and Decide phase are effected, e.g., controllable parameters like T.CW or M.CTW are modified to optimize the network performance.

One of the key benefits of this cognitive network node architecture, that significantly helps tactical communication, is its ability to represent the cognitive information gathered by a node's protocol stack in a compact fashion that can be easily exchanged with other nodes in the network. That is, our BN model captures the essence of the past network experiences of a user, tagged with spatial information, in a BN which can be stored in a central repository or shared with other tactical network nodes [2]. As an example of a simple application scenario, a soldier's radio device may gather valuable protocol stack behavioral information which, if shared with other soldiers' radio devices in the same area, may help them achieve a better network experience, due to the fact that the new devices can begin their operation with a near optimal protocol stack configuration, thus avoiding an expensive learning curve. Our cognitive network architecture is distributed and lightweight and, therefore, can be implemented even in resource constrained nodes. However, note that each BN applies only to a particular location.

In the next section, we describe the BN framework for designing the probabilistic structure between the network parameters, that will be the tool we use to perform the Learn phase of the cognition cycle. Then in Section III-B we describe how to exploit such probabilistic structure to design a proper estimator for the inference engine, performing the Plan and Decide phase of the cognition cycle.

III. Bayesian Networks preliminaries

A Bayesian Network (BN) is a graphical model for representing conditional independence relations between variables through a Directed Acyclic Graph (DAG). The BN model, well studied in machine learning [3], will be exploited in the Learn phase of our cognitive network to study the dependence structure that exists between the parameters of interest. This model was described in [4] and is summarized here for the reader's convenience. In general, given M discrete variables, $x_1, \ldots, x_M$, with unknown dependence relations, we assume that N independent instances of the M variables are observed and collected into the dataset $\mathcal{D}_{N,M}$ of size $N \times M$. From such dataset, through a procedure called structure learning, we can represent qualitatively the dependence relations among the variables in a DAG, where each random variable is represented as a node. The presence of an arrow between two nodes represents a direct probabilistic relation between them, while if there is no arrow the probabilistic relation between the two nodes can only be expressed through the other nodes in the path that connects them, according to the *d-separation* rules, e.g., see [3], [6]. In particular, we define a node i as a parent of another node h if there exists a direct edge from i to h and we write $i \in \text{pa}_h$, where pa_h is the set of parents of node h. Given the DAG we can also learn the quantitative relation between the parameters from the dataset $\mathcal{D}_{N,M}$ in linear time through parameter learning. Both structure learning and parameter learning are tools from machine learning [3] that have been applied to a vast range of problems e.g., medical diagnosis, risk factor analysis, terrorism risk management, reliability analysis of systems or enhancing human cognition [7].

A. Learning the BN: Structure Learning

Structure learning is the procedure to construct the DAG that represents the probabilistic relation between the random variables. In the literature there are two methods to design such DAG [6]: *a)* the constraint based method, in which a set of conditional independence statements is established, based on some *a priori* knowledge, to design the DAG following the rules of *d-separation*; and *b)* the score based method, commonly used in the absence of a set of given conditional independence

statements. We adopt the latter method, that is able to infer a suboptimal DAG from a sufficiently large dataset $\mathcal{D}_{N,M}$, and consists of two parts: 1) a search procedure to select a subset of the set containing all the DAGs; and 2) a function to score each DAG in the subset based on how accurately it represents the probabilistic relations between the variables in the dataset. The former is necessary since it is not computationally tractable to score all the possible DAGs given a set of M random variables, unless M is very small ($M \leq 5$), as an exhaustive enumeration of all structures ends to a number of possible DAGs that increases super exponentially with M. For this reason, it is necessary to define a search procedure that selects a small representative subset of the space of all DAGs, like the search strategies detailed in Section IV-C. The latter step consists in giving a score to each selected DAG and choosing the one with the highest score. 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.

In this paper we have chosen the Bayesian Information Criterion (BIC) as a score function. BIC is easy to compute and is based on the maximum likelihood criterion, i.e., how well the data suits a given structure, and penalizes DAGs with a larger number of edges. In the case in which all the variables are multinomial, with a finite set of outcomes r_i for each variable x_i, the general formula of BIC [8] can be reduced to a counting problem, as explained in the following. We define q_i as the number of configurations over the parents of x_i in the DAG S, *i.e.,* the number of different combinations of outcomes for the parents of x_i. We define also N_{ijl} as the number of outcomes of type l in the dataset $\mathcal{D}$ 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 $\mathcal{D}$ with parent configuration j. Given these definitions, it is possible to rewrite the BIC for multinomial variables as [9]:

$$\mathrm{BIC}(S|\mathcal{D}) = \sum_{i=1}^{M}\sum_{j=1}^{q_i}\sum_{l=1}^{r_i}\log_2\left(\frac{N_{ijl}}{N_{ij}}\right) - \frac{\log_2 N}{2}\sum_{i=1}^{M} q_i(r_i - 1) \ , \tag{1}$$

which is computationally tractable. A detailed comparison of scoring functions for structure learning is reported in [10].

B. Inference with the BN: Parameter Learning

In the third phase of the cognition cycle, the Plan and Decide phase, the cognitive network node should be able to infer new values for the controllable parameters of the network based on the observed parameters and the target values of the parameters that represent the performance, e.g., TCP throughput. To this aim we proposed the inference engine obtained exploiting the BN structure learned from the collected dataset $\mathcal{D}_{N,M}$. To build the inference engine it is necessary to calculate also the quantitative relations between the random variables of the BN, through a process called parameter learning, that consists in estimating the best set of parameters describing the conditional dependencies between the variables, given the independence relations defined by the DAG. According to the definition of BN, each variable is directly determined 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.

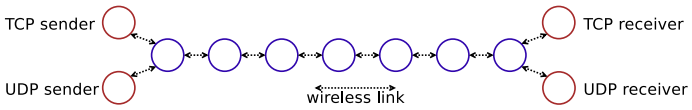

Fig. 2. Network scenario: dumbbell topology.

In this paper we choose the Maximum Likelihood Estimation (MLE) method for parameter learning, coherently with the choice of BIC as a scoring function for the structure learning algorithm [6]. In the case of multinomial variables, we can write the MLE as:

$$\hat{\theta}_{x_i=l|\mathrm{pa}_i=j} = \frac{N_{ijl}}{N_{ij}} \simeq P\left[x_i = l|\mathrm{pa}_i = j\right] \ , \tag{2}$$

where $\mathrm{pa}_i(j)$ for every j denotes a specific configuration, i.e., a set of values taken by the parents of node i. After inferring the quantitative relations through parameter learning, we can exploit the inference engine to predict the value of a chosen random variable x_i given the observation of a subset X_e of the other variables in the BN, called *evidence*. The inference engine uses Eq. (2) to calculate the probability mass function (pmf) of x_i given the evidence and based on this pmf it estimates the expected value of x_i at the discrete time k:

$$\hat{x}_i^{(k)} = E\left[x_i^{(k)}|X_e\right] \ . \tag{3}$$

IV. Inferring Network Congestion through a Bayesian Network Model of the Protocol Stack

In this section, we derive Bayesian Network (BN) structures that probabilistically connect some of the significant protocol stack parameters in different multi-hop wireless network scenarios. Then with the BN structure we design an engine that can be implemented in the cognitive network node to predict in advance the congestion status of the network. Knowing when congestion will arise is very useful for the efficiency of transport layer protocols; however, TCP and its popular variants do not have any mechanism to predict congestion in advance. Such a reactive nature of TCP leads to packet losses and wastage of precious network resources like bandwidth and energy, which are essential for efficient communication in mobile network environments such as tactical networks. In this paper, with the help of the BN structure derived observing the network environment and the current network state, we infer the congestion status of the network that will help TCP to proactively make decisions on how to adapt the value of the congestion window.

A. Network scenarios

We considered two multi-hop network topologies, a dumbbell topology depicted in Fig. 2 and a random topology, and we conducted experiments using the ns-3 discrete event network simulator [11]. The protocol stack in the simulator is augmented with hooks that collect the values of selected TCP and MAC parameters at regular sampling intervals for each flow. These hooks simulate the behavior of the Cognitive Agents (CA), that are responsible for reading and collecting the parameters' values. In the dumbbell topology, we have two nodes on either

side connected by a string of 7 intermediate nodes. Adjacent nodes are separated by 250 m, have a transmission range of 300 m, and have fixed PHY data rate of 2 Mbps. The Optimized Link State Routing (OLSR) is employed as the routing protocol. We have one TCP flow (*ftp* file transfer session from TCP sender to TCP receiver in Fig. 2) and one Constant Bit Rate (CBR) flow (UDP sender to UDP receiver in Fig. 2), hence each flow has eight hops. The CBR sending rate is equal to either 20 Kbps or 40 Kbps, to provide two levels of congestion to the TCP flow, Low Congestion (LC) and High Congestion (HC), respectively. The cognitive TCP source node samples the parameters of interest at 100 ms and 200 ms interval, respectively, in two sets of experiments. This scenario is very simple but sufficient to understand the basic relations between parameters.

In the second scenario, we have a mesh network with 40 nodes, initially arranged in an 8×5 pattern with 250 m between horizontally and vertically adjacent nodes and moving randomly at a speed of 1 m/s within a square area of 2500×2500 m^2. Each node has a transmission range of 300 m and is set up to be either the sender or the receiver in an *ftp* session over TCP. We have 20 such TCP flows that go on throughout the duration of the experiment. We have 20 of the nodes that were also involved as either transmitter or receiver of CBR traffic with rate 20 Kbps. The routing protocol is again OLSR and all the nodes were set to use the Minstrel physical layer rate control algorithm [12]. One of the *ftp* sender nodes was cognitive and capable of observing its network stack parameters at 100 ms intervals.

B. Network parameters

In this paper, we deal with MAC and TCP parameters, but the approach can be generalized to the collected parameters of the whole protocol stack. IEEE 802.11 and TCP are chosen as MAC and transport protocols, respectively. In each sampling interval k, the MAC parameters are the total number of original MAC packets transmitted in the sampling interval (M.TX(k), observable), the value of the 802.11 contention window at time k, i.e., the maximum number of slots the node will wait before transmitting a packet at the MAC level (M.CTW(k), controllable), and the total number of MAC retransmissions in the sampling interval (M.RTX(k), observable). The TCP parameters are the value of the Congestion Window at time k [bytes] (T.CW(k), controllable), the Round Trip Time, i.e., the last value registered in the sampling interval k of the time between when a packet is sent and when the corresponding acknowledgement is received (T.RTT(k), observable), the instantaneous TCP throughput, i.e., the total amount of unique data [bytes] acknowledged in a sampling interval k, divided by the length of the sampling interval [s] (T.TH(k), observable) and the network congestion status (T.CS(k), observable), which is a binary parameter equal to 1 when congestion is present and to 0 otherwise, defined as:

$$\text{T.CS}(k) = \begin{cases} 1\,, & \text{if T.CW}(k)/\text{T.CW}(k-1) \leq 0.6\,, \\ 0\,, & \text{if T.CW}(k)/\text{T.CW}(k-1) > 0.6\,. \end{cases} \quad (4)$$

The threshold is set to 0.6 because we are mainly interested in detecting significant drops of the congestion window. We aim to infer at time k the value of T.CS($k+n$), with $n = 1, \ldots, 5$, using current values of all the observable parameters, so we want to predict the occurrence of congestion to be able to act before it strongly affects the network.

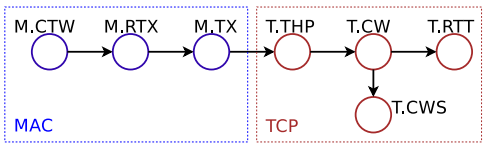

Fig. 3. BN structure learned from the historical network dataset and chosen as representative for all the multi-hop scenarios considered.

All the parameters collected, except T.RTT, are multinomial, with a finite but possibly large number of outcomes. In order to make the calculation simpler and more efficient we quantize all the parameters to a maximum of $n_q = 30$ levels, so it is possible to apply the learning algorithms for multinomial variables explained in Sections III and III-B, without the need for a very long training set to learn the probabilistic structure. Indeed, a finer quantization would lead to a more accurate estimation, but at the price of requiring a longer training set for proper learning of the probabilistic structure. Moreover, it was not realistic to introduce a complex non-uniform quantizer, given the limited computation capacity of the CA, so we chose to quantize the parameters' values according to their estimated n_q-quantiles, that translates in our case into better estimation performance than uniform quantization.

C. BN Structure Learning

The BN structure learning phase is the most computationally demanding, as for $M = 7$ parameters the search space contains 1.1×10^9 DAGs. We have used three search heuristics, namely *1)* Hill Climbing (HC) [13], *2)* a Markov Chain Monte Carlo method (MCMC) [13], and *3)* the simple ad hoc heuristic proposed in [4]. The first two heuristics are available in the MATLAB BNT toolbox [14]. We also implemented our heuristic in MATLAB which basically divides the structure learning problem into two sub problems, one for the MAC parameters ($M = 3$) and one for the TCP parameters ($M = 4$). It first finds the best BN structure for each one of them separately scoring all the possible DAGs, then gives these two separate BNs as the initial structure to the HC algorithm. Each of these three heuristics gives a DAG as a result and we choose among these three DAGs the one with the highest BIC score in Eq. (1). For each network scenario we obtain a slightly different DAG, and we choose the DAG in Fig. 3 as a structure for the inference in the rest of the paper, since it has a good score in all the scenarios considered, even if not optimal.

D. Congestion Inference

The prediction of T.CS, the congestion status, presents some fundamental issues that are addressed in this section. First of all, the BN gives us the qualitative (the DAG) and quantitative (Eq. (2)) probabilistic relations between the network parameters at a given time instant. In order to predict at time k the value of T.CS($k+n$), with $n \geq 1$, we need to introduce the time dimension in our model. In order to do so, we put the variable T.CS($k+n$) in our BN structure in place of T.CS(k) and we use the ML estimation in Eq. (2) to calculate the quantitative relations among the parameters of the new BN structure.

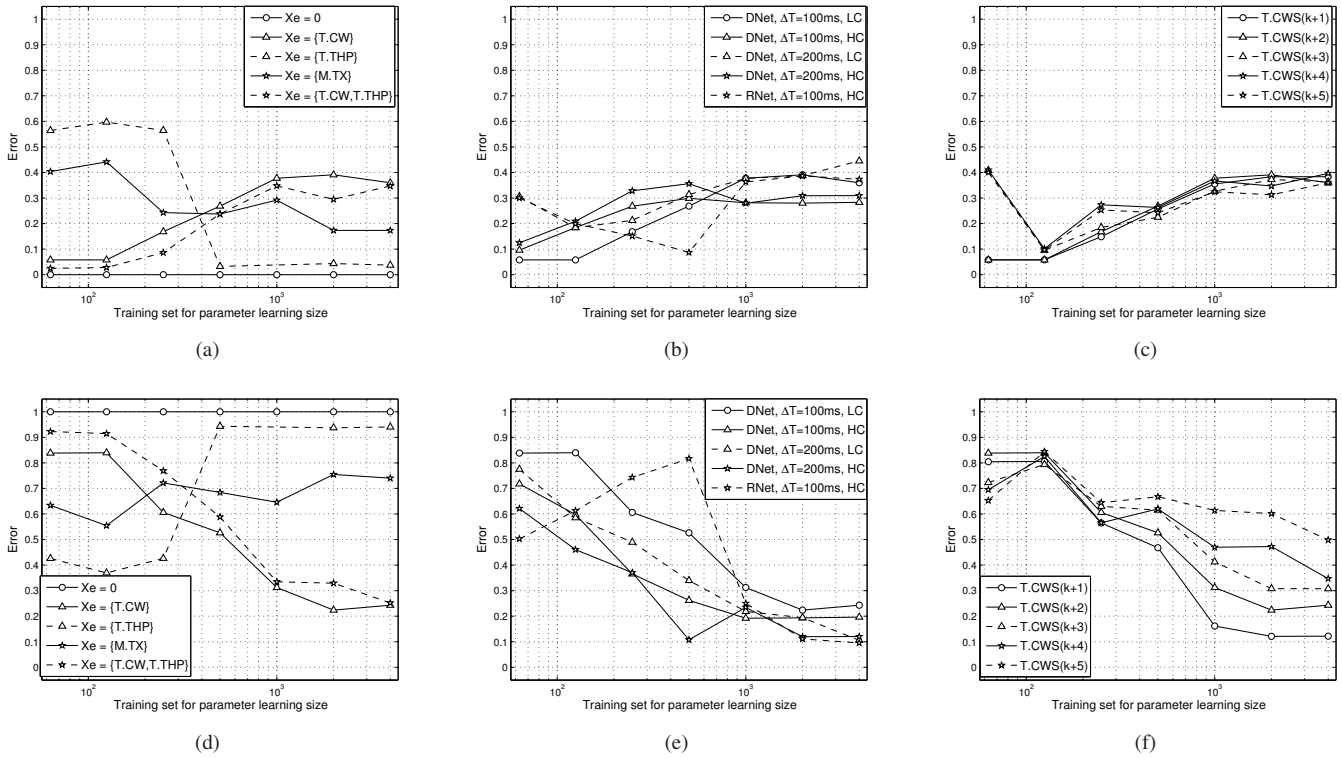

Fig. 4. Average prediction error for T.CS as a function of the training set length, when the value we aim to predict is T.CS$=0$ in (a), (b), (c) and when T.CS$=1$ in (d), (e), (f). In cases (a) and (d) we infer T.CS$(k+2)$ for different evidence sets. In cases (b) and (e) we infer T.CS$(k+2)$ for different network topologies (dumbbell network (DNet) and a random mobile network (RNet)), for different values of the sampling time ΔT and congestion levels (LC: low congestion and HC: high congestion). In cases (c) and (f) we infer T.CS$(k+n)$ for different values of n.

A second issue that arises is due to the fact that the prior for the variable T.CS$(k+n)$ is not uniform, but instead $P[\text{T.CW}(k+n)=0] \gg P[\text{T.CW}(k+n)=1]$, since congestion rarely occurs. If we simply aim at minimizing the misclassification rate, the predictor would always infer a value of $\widehat{\text{T.CS}}(k+n)=0$, because this is by far the most likely outcome. This predictor brings no information, so it is not useful. In order to solve this problem we introduce a loss function [3], i.e., we penalize with different weights the misclassification when congestion occurs, $L_{1,0}$, and when congestion does not occur, $L_{0,1}$. In other words, with the introduction of the loss function we aim at maximizing the probability of a certain occurrence multiplied by the corresponding loss function weight, i.e.:

$$P\left[\text{T.CS}(k+n)=b|\text{evidence}\right] \cdot L_{b,1-b}\ , \tag{5}$$

for $b \in \{0,1\}$. As we are interested in the relative values of these weights, we fix $L_{0,1}=1$ and we vary $L_{1,0}$. $L_{1,0}=1$ corresponds to the case in which we just aim at minimizing the overall misclassification rate. The inference engine we propose uses the probabilities in Eq. (2) to maximize Eq. (5) and as a result the inferred parameter given the evidence X_e is:

$$\widehat{\text{T.CS}}(k+n) = \max_{b\in\{0,1\}} P\left[\text{T.CS}(k+n)=b|X_e\right] \cdot L_{b,1-b}\ . \tag{6}$$

V. Performance analysis

In this section, we analyze the accuracy of the inference engine in predicting at time k the value of T.CS$(k+n)$, i.e., the presence or absence of congestion at time $k+n$, with $n \geq 1$. The performance of the engine is analyzed as a function of the length (in number of samples) of the training set used to learn the relations between the parameters and to define the inference engine with Eqs. (2) and (6). During the training set the parameters are recorded and then they become the input for the inference engine. In the y-axis of the figures we represent the average error for the inference, that is the expected value of $|\text{T.CS}(k+n) - \widehat{\text{T.CS}}(k+n)|$, where T.CS$(k+n)$ is the actual value of T.CS at time $k+n$ and $\widehat{\text{T.CS}}(k+n)$ is the inferred value. Since T.CS is a binary variable, this value can be viewed also as the frequency of an error in the estimation. We need to analyze separately the two cases in which the value we aim to infer is T.CS$(k+n)=0$ and T.CS$(k+n)=1$, since otherwise the average error would be dominated by the error in the former case (no congestion), that is a much more frequent event, and the predictor that minimizes the total error would simply give $\widehat{\text{T.CS}}(k+n)=0$, as discussed earlier.

In Fig. 4, we vary the evidence set, the network scenario and the value of n, fixing the value of the loss function to $L_{1,0}=7$, and we analyze the performance of the inference in case the value we aim to infer is T.CS$(k+n)=0$ (no congestion), in (a), (b) and (c), and T.CS$(k+n)=1$ (congestion), in (d), (e), and (f). In Figs. 4-(a)-(d) we represent the average error for the inference of T.CS$(k+2)$ in the dumbbell topology scenario of Fig. 2, in the case of low congestion (LC) and we vary the

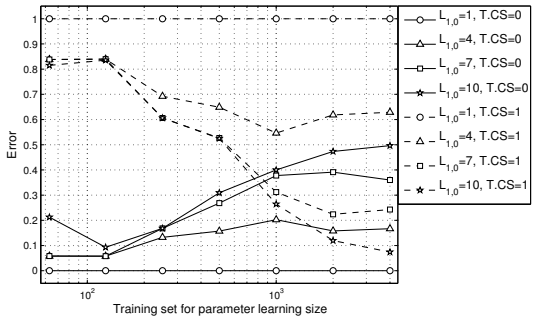

Fig. 5. Average prediction error for T.CS$(k+2)$ as a function of the training set length for different values of the loss function weight $L_{1,0}$, in case the value we want to infer is T.CS$(k+2)=0$ or in case T.CS$(k+2)=1$.

evidence set. In the case of no evidence, the prediction is equal to the prior for T.CS, so we always predict T.CS$(k+2)=0$, indeed in this case the error in Fig. 4-(a) is equal to zero, while in Fig. 4-(d) the error is equal to one, the maximum. The only predictors that give us useful information are the ones with T.CW in the evidence set. For a sufficient length of the training set, they give an error of almost 0.25 and 0.35, for congestion and no congestion (false positive), respectively.

In Figs. 4-(b)-(e) we consider as evidence T.CW(k) and compare the performance for different multi-hop network scenarios, i.e., the dumbbell topology in Fig. 2 with low congestion (LC) and high congestion (HC) and with the parameters sampled every $\Delta T = 100$ ms and $\Delta T = 200$ ms, and the random multi-hop network with mobility described in Section IV-A. The results depicted in the figure show that the inference engine performs similarly in these cases, with a frequency of false positives between 0.3 and 0.45 and a misclassification rate in case of congestion between 0.1 and 0.25 in all cases. Accordingly, we expect the inference engine to work well also in different kinds of topologies with different environmental conditions.

In Figs. 4-(c)-(f) we compare the performance of the inference engine for T.CS$(k+n)$, varying the value of $n = 1, \ldots, 5$, where k is the time sample at which we make the prediction and $k+n$ is the time sample of the predicted value $\widehat{\text{T.CS}}(k+n)$. We observe that the performance is almost constant when the actual value to be inferred is T.CS$(k+n)=0$, as shown in Fig. 4-(c), while it varies significantly when T.CS$(k+n)=1$. In this case, as expected, the performance decreases with n, and for $n=5$ we have a misclassification rate in case of congestion and in the absence of congestion of about 0.5 and 0.4, respectively, a performance close to the extreme case of the random predictor, and this gives an approximate limit in time to the possibility of predicting congestion with these models.

In Fig. 5 we vary the value of the loss function weight to determine a suitable value, considering the evidence $X_e =$T.CW(k). In case $L_{1,0}=1$ the inference engine is just predicting the most probable value, $\widehat{\text{T.CS}}(k+n)=0$. A good choice for $L_{1,0}$ is the one that guarantees a false positive error significantly smaller than 0.5 and minimizes the error in case of congestion, i.e., $L_{1,0}=7$. Furthermore, Fig. 5, in case T.CS$(k+n)=0$, may seem misleading since the average error grows with longer training set lengths, but this can be explained observing that, for a short training set, $N < 2 \cdot 10^2$, we have a large misclassification error (> 0.7) in case of congestion, so the predicted value is almost always $\widehat{\text{T.CS}}(k+n)=0$ in this case.

VI. Conclusions

In this paper, we proposed a cognitive node architecture for efficient communication in multi-hop wireless networks, that can significantly help in dynamic and challenging scenarios like tactical networks. We realized key functional modules of this architecture with the help of Bayesian Network models. We gathered values of significant parameters from the MAC and Transport layers in order to derive critical causality relations among these parameters in different network scenarios. As an interesting application of BNs for cognitive wireless networks, we studied the problem of predicting in advance the congestion status of the network. From the BN structure derived from network samples collected in the ns-3 simulation platform, we found that the current value of TCP's congestion window has a strong influence on accurately predicting the congestion status of the network at future time instants and we analyzed the performance of such inference engine. Future work involves devising algorithms that exploit the predicted congestion status of the network to adapt the value of the congestion window before a congestion occurs and studying the effect of misclassification rates on TCP performance in terms of throughput and packet losses.

Acknowledgments

This work was partially supported by the U.S. Army Research Office, Grant. No. W911NF-09-1-0456 and by 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] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] G. Quer, H. Meenakshisundaram, B. R. Tamma, B. S. Manoj, R. Rao, and M. Zorzi, "Cognitive Network Inference through Bayesian Network Analysis," in *IEEE Globecom*, Miami, FL, US, Dec. 2010.

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

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

[7] O. Pourret, P. Naim, and B. Marcot, *Bayesian Networks: A practical guide to Application*. Wiley, 2008.

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

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

[10] 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.

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

[12] "Minstrel physical layer rate adaptation algorithm," Last time accessed: April 2010. [Online]. Available: madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt

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

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