Topology Control Scheduling

Based on the Distributed Learning Automata

Mina Shirali
Member of Young Researchers Club
Islamic Azad University of Qazvin
Qazvin, Iran
m.shirali@qiau.ac.ir

Mohammad Reza Meybodi
Computer Engineering Department
Amirkabir University of Technology
Tehran, Iran
mmeybodi@aut.ac.ir

Hamid Daneshvar Tarigh
Member of Young Researchers Club
Islamic Azad University of Qazvin
Qazvin, Iran
daneshvar_tarigh@qiau.ac.ir

***Abstract*—This paper, presents a learning automata based algorithm (LABTA) to determine an appropriate time interval for topology control, while capturing network updates and minimizing algorithm overhead. This algorithm is application dependent and it is possible to consider other parameters like delay, etc. In this paper, we describe how to merge parameters and compute the fitness. We have implemented the proposed algorithm with both of standard and distributed models of the learning automata. Simulation results show that our proposed algorithm outperforms periodic topology control. In addition, a comparison can be also found between two most well known topology control protocols; LMST and K-Neigh.**

Keywords-component; topology control; learning automata; scheduling; hello interval

I. INTRODUCTION

Topology Control (TC) maintains a topology with certain properties (e.g., connectivity) while reducing energy consumption and/or increasing network capacity [1]. The importance of topology control lies in the fact that it critically affects the system performance in several ways. For example, as shown in [2], it affects network spatial reuse and hence the traffic carrying capacity.

The time interval between periodical topology controls is called hello interval. In the dynamic networks with mobile nodes, a short hello interval is necessary to compensate the frequent topology changes, which can lead to the vase power consumption and more overhead. Thus, this parameter must be tuned precisely.

This paper presents a localized learning automata based algorithm (LABTA) to determine the hello interval of topology control. As our simulation results show, LABTA outperforms periodical topology control. Two well known algorithms (LMST and K-Neigh) are used as topology control protocols [3, 4]. Therefore, a comparison can be also found between them.

The rest of this paper is organized as follows: In section 2 some well known topology control algorithms are introduced. Also, impact of hello interval on the performance of topology control is described in this section. Section 4, represents a pseudo code of LABTA. Simulation results are presented in the section 5, and finally conclusion is drawn in the section 6.

II. RELATED WORKS

Many works are done on topology control in the adhoc networks. In the following, we introduce some well known of them. The MST which is the most well known between heuristic algorithms, defines the smallest subset of edges that keeps the graph in one connected component [3]. Unfortunately, MST can not be computed locally. In LMST [3], each node builds its local minimal spanning tree. In fact, LMST is a distributed version of MST.

K-Neigh [4] is a fully distributed, asynchronous, and localized protocol, based on the number of neighbors that each node should have in order to achieve full connectivity with a high probability. In LINT [5], each node periodically checks number of active neighbors and changes its power level accordingly, so that the node degree is kept within a threshold. In [6], two topology control algorithms are presented: ABsolute Distance-based (ABD) and PRedictive Distance-based (PRD). These algorithms, also, attempt to maintain a logical number of neighbors between predefined values K*min* and K*max*.

A distributed topology control protocol called CBTC (Cone Based Topology Control) is proposed in [7], which is based on the directional information. Basic idea is that, each node gradually increases its transmission power until it finds a neighbor node in every direction (cone). As the result, the global connectivity is guaranteed with minimum power for each node. Compared with CBTC, the K-Neigh protocol relies on a weaker assumption, i.e., distance estimation vs. directional information. Furthermore, CBTC has no bound on the number of messages nor on the energy expended in determining the proper transmit power, whereas in the K-Neigh protocol each node transmits only two messages at a predefined power (the maximum transmit power). Simulation results of [4] show that the topologies generated by the K-Neigh protocol are (on the average) 20% more energy efficient than those generated by CBTC.

As mentioned before, the time interval between periodical topology controls is called hello interval. A similar case can be seen in the routing, where neighbor nodes exchange information with each other by piggybacking in the periodically broadcast “Hello” messages. Some works are also done on the hello interval of routing algorithms [8, 9, 10].

978-1-4244-3709-2/10/$25.00 ©2010 IEEE

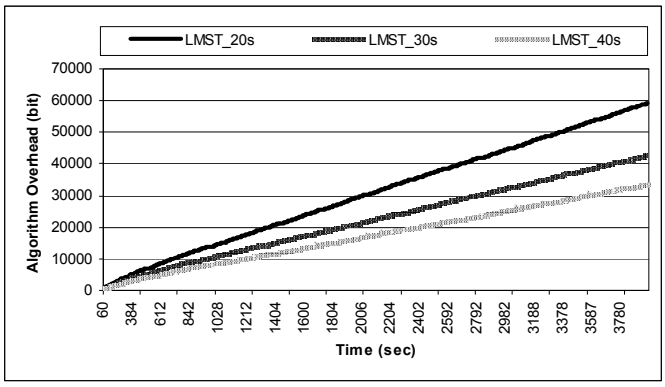

Figure 1. Impact of hello interval on the algorithm overhead

In the high-density networks with fast mobility, the change rate of topology is relatively high. Since topology changes are not advertised until next hello messages broadcast, under such circumstances, topology changes might be too fast to be captured by periodic updates. Therefore, a short hello interval is necessary to compensate the frequent topology changes. However, a short ‘‘Hello” interval leads to frequently execution of the protocol, and consequently leads to the vase power consumption [11]. Thus, this parameter must be tuned precisely. For example, impact of hello interval on the algorithm overhead is shown in the Fig. 1. In this figure, we have used LMST as topology control algorithm. However, impact of hello interval on the performance of topology control depends on a range of network parameters including node density and node velocity. Accordingly, reducing these intervals may not lead to performance improvements [8]. In [3], the time interval between two broadcasts of hello messages depends on the level of node mobility, and is determined by a probabilistic model. Also, in [11], K-Neigh protocol is analyzed through a probabilistic model, for different spatial and temporal properties such as hello interval, logical node degree and final transmission range. Unfortunately, to the best of our knowledge, just a few works are done on this area, and none of them have used meta-heuristics like GA, NN and learning automata (LA). In this paper, a localized algorithm (LABTA) is proposed, that uses learning automata to determine the next topology control interval in the mobile adhoc networks.

III. LABTA

As mentioned before, impact of hello interval on the performance of topology control depends on a range of parameters, and it seems difficult to determine it through a polynomial statement, in a mobile adhoc network. Therefore, our algorithm is designed heuristically. LABTA uses learning automata to determine the next hello interval. In the following, we describe how to compute fitness of an action. Fitness factor of i*th* node in d*th* step can be computed as (1):

$$f_{di} = \frac{\prod_{j=1}^{m}(\rho_{ij}\Phi(u_{dij}) + \overline{\rho_{ij}}) - \prod_{j=1}^{m}\overline{\rho_{ij}}}{1 - \prod_{j=1}^{m}\overline{\rho_{ij}}} \quad \forall 0 \le \rho_{ij}, \Psi, \Phi \le 1 \tag{1}$$

$$0 \le \rho_{ij} \le 1 \quad \Rightarrow \quad \overline{\rho_{ij}} = 1 - \rho_{ij} \tag{2}$$

Where, m is number of utilization metrics, u_{dij} is the j*th* utilization metric of the i*th* node in d*th* step, $\Phi(u_{dij})$ is the normalized value of u_{dij}, and ρ_{ij} is the effect constant of u_{dij}. Our goal is to propose a simple algorithm that maximize the hello interval to minimize algorithm overhead, while capturing network updates. For this purpose, we have used two metrics as below:

- Variation in the power level (u_{div})
- Algorithm overhead (u_{dio})

However, with the proposed fitness function, some other parameters (like number of dropped packets, variation in the number of neighbors, etc) can be considered, too. Now let u_{dio}, u_{dip} and u_{div} be the algorithm overhead, power level and variation in the power level of the i*th* node in the d*th* step. Also, let ρ_{io} and ρ_{iv} be the related effect constants of u_{dio} and u_{div}, respectively. u_{div} is computed as (3):

$$u_{div} = \left| u_{dip}^{Mean} - u_{dip} \right| \tag{3}$$

Where, u_{dip}^{Mean} is the mean value of u_{dip} through d steps. Since our metrics have to decrease, we normalize them with (4). $u_{dij}^{\min}$ and $u_{dij}^{\max}$ are the minimum and maximum threshold of u_{dij}, respectively.

$$\phi(u_{dij}) \begin{cases} \dfrac{u_{dij}^{\max} - u_{dij}}{u_{dij}^{\max} - u_{dij}^{\min}} & u_{dij}^{\min} \le u_{dij} \le u_{dij}^{\max} \\ 0 & u_{dij} > u_{dij}^{\max} \\ 1 & u_{dij} < u_{dij}^{\min} \end{cases} \tag{4}$$

In our system, $u_{dij}^{\min}$ and $u_{dij}^{\max}$ are updated dynamically. For example, the maximum amount of power variation obtained through d steps is computed as (5):

$$u_{div}^{\max} = \max\{u_{div}^{\max}, u_{liv}, 0\} \quad \forall 1 \le l \le d \tag{5}$$

That is, at first $u_{div}^{\max}$ is zero. Then, at instance *d*, if $u_{div}^{\max}$ is less than u_{div}, set it as u_{div}. Otherwise, keep it unchanged. This $u_{div}^{\max}$ is not the actual value of maximum threshold, but provides an overestimate value. Alternatively, we could choose it as a large constant. But as one can imagine, a large maximum threshold causes a significant decrease in the convergence speed. In addition, $\Phi(u_{div})$ becomes a small value and rounding errors can have significant impacts on the system performance, consequently. The polynomial statement of the fitness is computed as (7):

$$fitness_{di} = \frac{c}{\sum_{k=1} r_{ik}} \sum_{k=1}^{f} r_{ik} f_{dik} \tag{6}$$

Where *f* is number of factors, *c* is a positive real factor, f_{dik} is k*th* factor of the i*th* node in the d*th* step, and $0 \le r_{ik} \le 1$ is the importance constant of k*th* factor. However, in this paper number of factors is one and our fitness factor is computed as (8):

$$fitness = r_{i1}\Phi(u_{dio})\Phi(u_{div}) \tag{7}$$

We can improve this fitness factor, using a variable effect constant as (9):

$$\begin{aligned} &\rho_{dij} = \overline{\Phi(u_{dij})} \\ &0 \le \Phi \le 1 \quad \Rightarrow \quad \overline{\Phi} = 1 - \Phi \end{aligned} \tag{8}$$

Therefore, using this effect constant, fitness statement changes as (10):

$$\begin{aligned} A &= (\Phi(u_{dio}) \times \overline{\Phi(u_{dio})} + \Phi(u_{dio})) \\ B &= (\Phi(u_{div}) \times \overline{\Phi(u_{div})} + \Phi(u_{div})) \\ fitness &= r_{i1} \frac{(A \times B) - \Phi(u_{dio}) \times \Phi(u_{div}))}{1 - \Phi(u_{dio}) \times \Phi(u_{div})} \end{aligned} \tag{9}$$

For simplicity, we abbreviate this type of fitness statement as (11):

$$fitness = (O)(V)\rho \tag{10}$$

Where *O* is overhead, *V* is power variation and 'ρ' implies that we have used described variable effect constant and fitness is computed as (10).

LABTA uses the continuous pursuit reward-penalty (**CP*RP***) [15] model of learning automata, which is consist of three steps: The first step consists of choosing an action based on the probability distribution. Whether the automaton is rewarded or penalized, the second step is to increase the component of whose reward estimate is maximal (the current optimal action), and to decrease the probability of all the other actions. The last step is to update the running estimates for the probability of being rewarded. For this purpose, it uses vector $\underline{D}$ to keep the probability of being rewarded for its actions. For more improvement, we can use fitness to compute $\underline{D}$ as (11):

$$\underline{D}_i(t+1) = \underline{D}_i(t) + fitness_{di} \tag{11}$$

Where, $\underline{D}_i(t+1)$ is the rewarding probability of i*th* action at instance t+1. Note that fitness is a real positive number that its maximum value is one.

Now we are to describe main algorithm of LABTA. Our proposed algorithm is implemented with both of the standard [12] and distributed [13, 14] models of learning automata. The standard model of automata is a distributed automata model with only one LA.

Note that our network has *n* nodes, each node has a DLA, and each DLA has *m* LA itself. That is, there are *n*×*m* LA. Therefore, there are *n* LA when we use the standard model. As mentioned before, we use **CP*RP*** to update the probability vector of each LA. A pseudo code of the proposed algorithm is given below:

1. ***Initialize***:
   1.1. *Set* d=*0*
   1.2. *Initialize probability vector* p_{1ij} *, for each LA*j.
   1.3. *Activate an automata randomly (LA*a*).*
2. d=d+*1*.
3. *LA*a *selects an action according to its* p_{dia} .
4. *the hello interval is determined according to the selected action.*
5. *At the time instance that is determined in the previous step, each node sends a hello message.*
6. *LA*a *computes the fitness of its selected action, according to the obtained information.*
7. *LA*a *updates its action probability, according to the* CP_{RP}.
8. *According to the selected action, corresponding automata is activated. That is, LA*a *changes.*
9. *If probability vector converges, stop. Otherwise, go to the step 2.*

Figure 2. LABTA in the ith node

IV. SIMULATION

We have used OPNET to simulate our network, which is a powerful simulator. Also, we have used its predefined settings for IEEE 802.11b standard model. For example, data rate is 11 Mbps and the bandwidth is 2.4 GHz. Signal propagation model is free space-line of sight, which is predefined in the OPNET. One of the basic features of a mobile adhoc network (MANET) is that nodes move according to some mobility model. For MANETs, the random way-point model (RWP) [10] is, by far, the most popular mobility model. The speed (m/s) and pause time(s) parameters of all nodes are set as uniform(0,10) and constant(1), respectively. However, in the static networks nodes don't have any mobility. Nodes are placed in a 500*500 meter area non-uniformly. Due to the massive amount of data to be processed, strictly a simulator issue, the number of nodes is limited to 15 and used as a proof-of concept. All nodes generate traffic for a random destination and the packet inter arrival time is set as Poisson(1). We have tested our proposed algorithm in both of the static and dynamic networks. An important specification of static networks is number of neighbors and network density. We assumed that at first each node has at least one neighbor. Simulation results are summarized in the TABLE I, where 'S' implies that we didn't use DLA, and 'B' implies that we didn't use fitness to compute probability of being rewarded in $\mathbf{CP}_{RP}$. Also, 'L' implies that LABTA is implemented with LMST, and 'K' implies that LABTA is implemented with K-Neigh. The "Dropped" column of this table determines number of bits that are dropped due to their exceeded retry threshold and, the "Throughput" column represents the average number of bits that are received by the receiver channel, successfully (in the mac layer of network).

TABLE I. SIMULATION RESULTS

	Rreceived (bit)	Received (bit/mW)	Dropped (bit)	Thrpt (bit)
LMST	12120064	13.069	8237896	92938972
L (O)(V)	12420096	13.388	8069088	101052916
L (O)(V) p	12574720	14.106	8560976	109284292
K-Neigh	19766272	12.887	9579288	136619632
K (O)(V)	20120576	14.299	9021944	145063236
K (O)(V) p	22137856	14.425	9775696	156603152
L (O)(V) B	11203584	13.192	8411752	90039520
L (O)(V) S	11587584	13.100	8894688	94872944
L (O)(V) BS	11071488	13.085	9506312	93764112

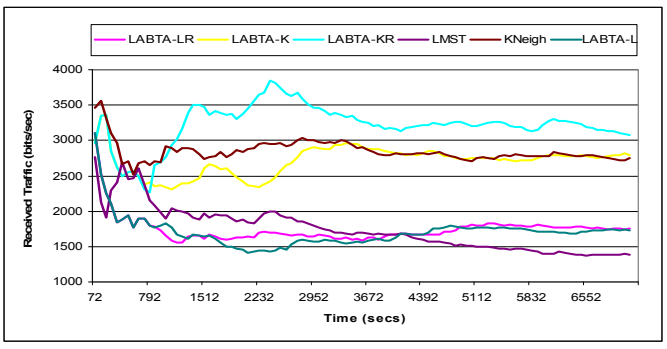

Figure 3. Average of received bits per second in LABTA

V. CONCLUSION

A distributed learning automata based algorithm is proposed in this paper, to determine an appropriate hello interval. The proposed algorithm (LABTA), is application dependent and it is possible to select metrics according to the related application. For example, if QOS is important, delay can be also considered as a metric. As our simulation results show, LABTA outperforms periodical topology control.

As mentioned before, LABTA is implemented with both of the standard and distributed models of learning automata. DLA considers a weighted graph of transitions between actions and provides more accurate decisions. Therefore, using DLA, better results can be provided. Also, using fitness we can provide continuous and more accurate values fitness to compute probability of being rewarded in $\mathbf{CP}_{RP}$, and better results can be provided, consequently. From our results, it is also obvious that, K-Neigh protocol, however, does not guarantee connectivity even though it does achieve connectivity with a high probability.

REFERENCES

[1] P.Santi, "Topology control in wireless ad hoc and sensor networks," John Wiley and Sons, Chichester, UK, July 2005.

[2] P. Gupta, and P. R. Kumar, "The capacity of wireless networks," IEEE Trans. Inform. Theory, vol. 46, pp. 388–404, March 2000.

[3] N. Li, J. C. Hou, and L. Sha, "Design and analysis of an MST-based topology control algorithm," IEEE Trans. Wireless Communications, vol. 4, pp. 1195–1206, May 2005.

[4] D.M. Blough, M. Leoncini, G. Resta, and P. Santi, "The K-Neigh protocol for symmetric topology control in ad hoc networks," in Proc. ACM MobiHoc 03., pp. 141–152, 2003.

[5] R. Ramanathan, and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in Proc. IEEE INFOCOM Conf., pp. 404–413, 2000.

[6] A. C.-C. Yao, "Mobility-aware topology control in mobile ad hoc networks," Elsevier J. Computer Communications, vol. 31, pp. 3521-3532, September 2008.

[7] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," in Proc. IEEE INFOCOM Conf., pp. 1388-1397, 2001.

[8] Y. Huang, S. Bhatti and S. Sorensen, "Self-tuning network support for MANETs," In Proc. IEEE NOMS2008 conf., Salvador, Bahia, Brazil, vol. 7-11, pp. 1037-1042, April 2008.

[9] M. Voorhaen and C. Blondia, "Analyzing the impact of neighbor sensing on the performance of the OLSR protocol," In Proc. WiOpt06 conf., Boston, Massachusetts, 2006.

[10] H. Tan, W. Zeng, and L. Bao, "PATM: Priority-Based Adaptive Topology Management for efficient routing in ad hoc networks," In Proc. Computational Science conf., pp. 485-492, 2005.

[11] A. Nayebi, and H. Sarbazi-Azad, "Analysis of K-Neigh topology control protocol for mobile wireless networks," In Proc. Elsevier Computer Networks conf., vol. 53, pp. 613-633, 2008.

[12] K. Narendra, and M. A. L. Thathachar, "Learning automata: An introduction," Englewood Cliffs, NJ: Prentice Hall, 1989.

[13] H. Beigy, and M. R. Meybodi, "A new distributed learning automata based algorithm for solving stochastic shortest path problem," In Proc. the Sixth International Joint Conference on Information Science, Durham, USA, pp. 339-343, 2002.

[14] R. Forsati, M. R. Meybodi, and M. Mahdavi, "Web page personalization based on distributed learning automata," In Proc. IKT, Ferdowsi University of Mashad, Mashad, Iran, November, 2007.

[15] M. A. L. Thathachar, and P. S. Sastry, "Estimator algorithms for learning automata," in Proc. Platinum Jubilee Conference on Systems and Signal Process, Bangalore, India, pp. 29-32, Dec. 1986.