Proceedings of the 4th National Conference; INDIACom-2010
Computing For Nation Development, February 25 – 26, 2010
Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

Validation of Clustering Algorithm for Mobile Ad-Hoc Networks Using Colored Petri Nets

Kranthi Kumar Amirineni
Department of Computer Science, NIT Rourkela, Rourkela-769008
Email: kranthi.amirineni@gmail.com
Suchismita Chinara
Department of Computer Science, NIT Rourkela, Rourkela-769008
Email: suchi.nitrkl@gmail.com
Santanu Kumar Rath
Department of Computer Science, NIT Rourkela, Rourkela-769008
Email: rath.santanu@gmail.com

ABSTRACT

Validation of protocols is essential to ensure that the protocol is unambiguous, complete and functionally correct. One approach to ensure the correctness of an existing protocol is to create a formal model for the protocol, and analyze the model to determine if indeed the protocol provides the defined services correctly. Due to the dynamic nature, concurrency and different levels of abstraction associated with the Mobile Ad-Hoc Network (MANET) protocols, it is difficult to design a model for MANET protocols with existing techniques like UML and other modeling languages. Colored Petri Nets (CPN) is a suitable modeling language for this purpose. This is a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic and stochastic. As a graphical tool, CPNs can be used as a visual-communication aid similar to flow charts, block diagrams and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of the system. This paper considers a weighted clustering algorithm (WCA) for MANETs which takes into consideration the ideal degree, transmission power, mobility and battery power of a mobile node. In MANET a node can move in and out and thus the MANETs topology (graph) dynamically changes. This paper considers a Topology Approximation (TA) mechanism to address this problem of mobility and perform simulations of typical protocol called On-Demand WCA. Our simulation reports validate the correctness of both TA mechanism in terms of dynamically changing topology of MANETs and WCA in terms of number of cluster heads and dominant set updates.

KEYWORDS

Colored Petri net (CPN), Mobile ad hoc network (MANET), Weighted clustering algorithm (WCA), Validation.

I. INTRODUCTION

A Multi-cluster, multi-hop packet radio network architecture for wireless systems should be able to dynamically adapt itself with the changing network configurations. Certain nodes, known as cluster heads, are responsible for the formation of clusters and maintenance of the topology of the network. The set of cluster heads is known as a dominant set. A cluster head does the resource allocation to all the nodes belonging to its cluster. Due to the dynamic nature of the mobile nodes, their association and dissociation with the clusters disturb the stability of the network and thus reconfiguration of cluster heads is unavoidable. This is an important issue since frequent cluster head changes adversely affect the performance of other protocols such as scheduling, routing and resource allocation that rely on it [1].

This paper considers a weighted clustering algorithm (WCA) which takes into account the number of nodes a cluster head can handle ideally (without any severe degradation of the system performance), transmission power, mobility and battery power of the nodes [2]. For successful application of MANETs, it is very important to ensure that the protocol is unambiguous, complete and functionally correct. One approach to ensure correctness of an existing protocol is to create a formal model for the protocol, and analyze the model to determine if indeed the protocol provides the defined services correctly. Colored Petri Nets (CPN) is suitable modeling language for this purpose, as it can conveniently express non-determinism, concurrency and different levels of abstraction that are inherent in protocols [3], [4].

CPNs provide several analysis methods, including simulation, state space analysis, sweep-line analysis and language analysis [3], [4]. In addition it has a good computer tool support, CPN Tool. CPN Tool [5] is a suite of tools for editing, simulating, analyzing CPNs and it has a graphical editor that allows the user to create and layout the different net components. One of its nice features is that it uses pages to visually divide the model into components, enhancing its maintainability and readability without affecting the execution or analysis of the model. However it is not easy to build a CPN model of a MANET because nodes can move in and out of their transmission ranges and thus MANETs topology dynamically changes. Currently there are few mechanisms proposed to model a system's dynamic structure using CPN. In this paper topology approximation mechanism [6] is used to address the problem of mobility in MANETs and perform simulation of a typical protocol called WCA. The TA mechanism describes the aggregate behavior of nodes where their long-term average behaviors are of interest.

The rest of this paper is organized as follows, section II discuses the Topology Approximation mechanism for mobility problem of a MANET in detail. Section III presents our CPN models of a MANET using an On-Demand WCA and the TA mechanism. Simulation results are given in section IV. Conclusion and future scope in section V.

II. MOBILITY PROBLEMS AND TOPOLOGY APPROXIMATION

In this paper assume that the MANET is fully symmetric, i.e., every node has identical capabilities and responsibilities. A MANET can be represented by a graph G(V,E), where V is the set of nodes representing mobile hosts, and E is the set of edges representing links interconnecting mobile hosts. In a MANET if node A lies within the transmission range of another node B, we say there is link between them in the graph describing the MANET, where node A is called a neighbor of node B, and vice versa. For example in Figure.1, nodes A and B are neighbors since there with in the transmission ranges of each other, but node C is not a neighbor of node A since not C is not within the transmission range of node A.

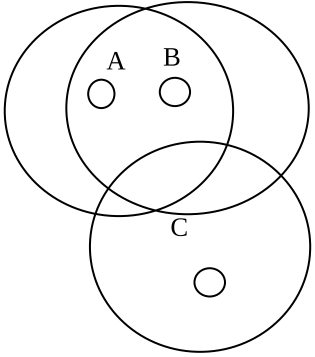

Figure.1

Every node is allowed to move at will in a MANET, and thus a link between any two nodes may disappear and reappear in an unpredictable manner. For example in Figure.1, when node A moves out of the transmission range of node B, the link between node A and B breaks as shown in the Figure.2. When node B moves back within the transmission range of node A, the link between them reappears as shown in the graph of Figure.3.Therefore it is difficult to build a dynamic structure in CPN model.

The TA mechanism uses a receiving function [6] to describe the dynamic structure of a MANET. This paper uses the general assumption in MANET that every node has the same distance of transmission range. Thus if a MANET is composed of a fixed number of nodes, the average number of neighbors of a node will depend on the area it covers.

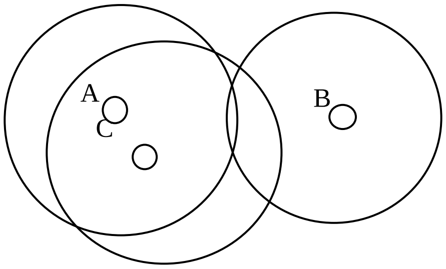

Figure.2

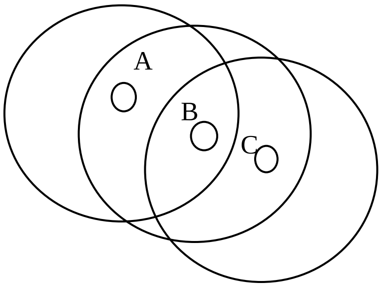

Figure.3

If all nodes dispersed in a small area, the average number of neighbors will be larger than that when neighbors are dispersed in a wider area. Thus average number of neighbors can well represent the approximate topology information of a MANET. Based on this observation, the TA mechanism uses a receiving function to decide which nodes receive which messages. In the graph describing a MANET, the average number of neighbors can be described by the average degree of nodes in the graph. Thus by using the average degree of the graph representing a MANET, the CPN model can provide enough information for finding neighbors [6].

Assume every node in a MANET has the same number of neighbors which is equal to the average degree of the MANET graph. Since every node has the same capacity and responsibility, every node in a MANET has the same chance of receiving or forwarding broadcasting messages. In CPN modeling, create a place called MsgStore to hold all the messages in transmission. After the place MsgStore receives the messages, a receiving function directs messages to the corresponding neighboring nodes. Clearly, a node's maximum number of neighbors is equal to (n-1) where n is the number of nodes in a MANET, and all the nodes neighbors will receive the broadcast message sent by the node. Thus a node should maximally send (n-1) copies of broadcast messages to the MsgStore. But only x out of these (n-1) copies of the broadcast message will be actually received by other nodes where x is the number of neighboring nodes of this node, and (n-1-x) copies will be thrown away by the CPN model. The function of how many copies of broadcast messages will be received by other nodes are achieved by the receiving function in using probability bar (PB). PB is the probability of a node that will receive a broadcast message. Let d be the average degree of the

MANET graph. Then on average, d nodes will receive a broadcast message among all (n-1)(n-1) broadcast messages. Thus we have

$$PB = \frac{d}{(n-1)(n-1)}$$

III. CPN MODEL OF A MANET WITH ON-DEMAND WCA

A. CPN MODEL OF A MANET

The hierarchy model of CPN for a MANET is depicted in Figure.4, and shows the over-all organization of the modules comprising the CPN model of MANET. Page MANET is the top level of CPN model for a MANET, which consists of ten nodes in this example. The second level is node template for implementing WCA. By instantiating this node template, any can create a CPN model for a MANET composed of nodes as many as allowed in the computer using CPN Tool.

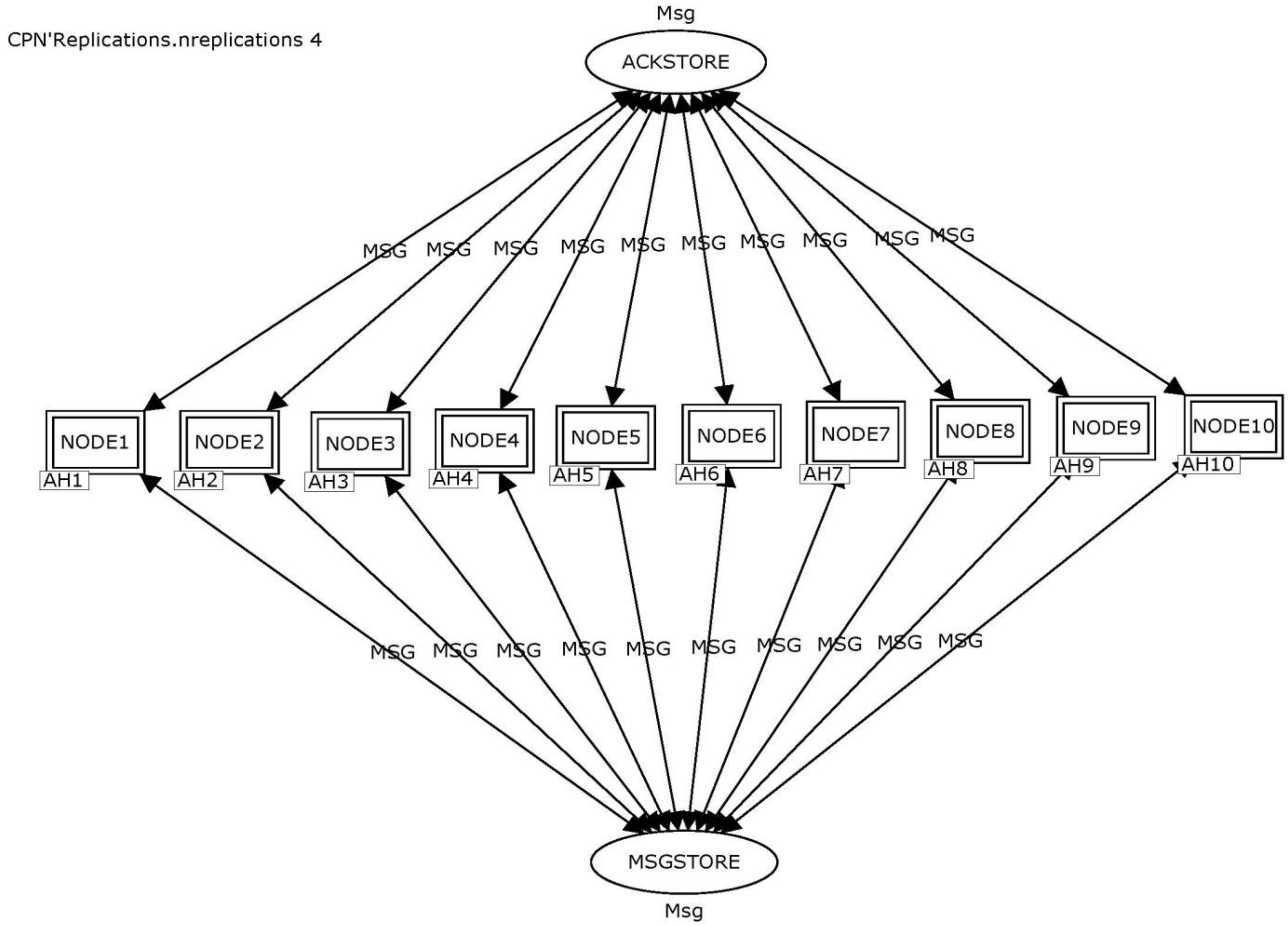

Figure.4 CPN Model of A MANET

The function of the node template is similar to that of class in an object-oriented programming language. If anybody wants to create a node instance in a MANET, simply instantiate the node template is sufficient. Like an object in an object-oriented programming language, every instantiated node has its own local variables and provides interface to the outside world [3], [4]. Here in this CPN modeling, an instantiated node is represented by a substitution transition, and sockets provided by CPN tool are its interfaces to the outside world. If more nodes are to be added to this MANET model, by simply add substitution transitions to the model. Figure.4 shows a node instance in a MANET, which is a part of the CPN model in the top level.

B. CPN MODEL OF ON-DEMAND WCA

As mentioned in the above, the second level is the node template implementing WCA. WCA is a typical algorithm which calculates the cluster heads of each and every node. To decide how well suited a node is to be a cluster head, take into account its degree, transmission power, mobility and battery power [2]. The following features are considered in the weighted clustering algorithm (WCA).

- The cluster head election procedure is not periodic and invoked as rarely as possible. This reduces system updates and hence computation and communication costs.

- Each cluster head can ideally support M (a pre-defined system threshold) nodes to ensure efficient MAC functioning. A high throughput of the system can be achieved by limiting or optimizing the number of nodes in each cluster.
- The battery power can be efficiently used within certain transmission range. Consumption of the battery power is more if a node acts as a cluster head rather than an ordinary node.
- Mobility is an important factor in deciding the cluster heads. Reaffiliation occurs when one of the ordinary nodes moves out of a cluster and joins another existing cluster. In this case, the amount of information exchange between the node and the corresponding cluster head is local and relatively small. The information update in the event of a change in the dominant set is much more than a reaffiliation.

CLUSTERHEAD ELECTION PROCEDURE

Step 1: Find the neighbors of each node v (i.e. nodes within its transmission range). This gives the degree d_v, of this node.

Step 2: Compute the degree-deference, $D_v = | d_v - M |$, for every node v.

Step 3: For every node, compute the sum of the distances, P_v, with all its neighbors.

Step 4: Compute the running average of the speed for every node. This gives a measure of mobility and is denoted by M_v.

Step 5: Compute the time, T_v, of a node v during which it acts as a cluster head. T_v indicates how much battery power has been consumed since we assumed that consumption of battery power is more for a cluster head than for an ordinary node.

Step 6: Calculate the combined weight $I_v = c_1 D_v + c_2 P_v + c_3M_v + c_4T_v$, for each node v. The coefficients c_1, c_2, c_3 and c_4 are the weighting factors for the corresponding system parameters.

Step 7: Choose the node with a minimum I_v to be the cluster head. All the neighbors of the chosen cluster head can no longer participate in the election algorithm.

Step 8: Repeat Steps 2 to 7 for the remaining nodes not at assigned to any cluster.

Every mobile node in any system (GSM or CDMA) periodically exchange control information with the base station. Similar idea is applied here, where all the nodes continuously monitor their signal strength as received from the cluster head. When the mutual separation between the node and its cluster head increases, the signal strength decreases. In that case, the mobile has to notify its current cluster head that it is no longer able to attach itself to that cluster head. The cluster head tries to handover the node to a neighboring cluster (existing cluster head in the dominant set), this process being called a reaffiliation. The cluster head of the reaffiliated node updates its member list. If the node goes into a region not covered by any cluster head, then the cluster head election algorithm is invoked and the new dominant set is obtained.

Figure.5 shows the template of a CPN node for implementing On-Demand WCA. It is represented by substitution transition AH1, in Figure.4.

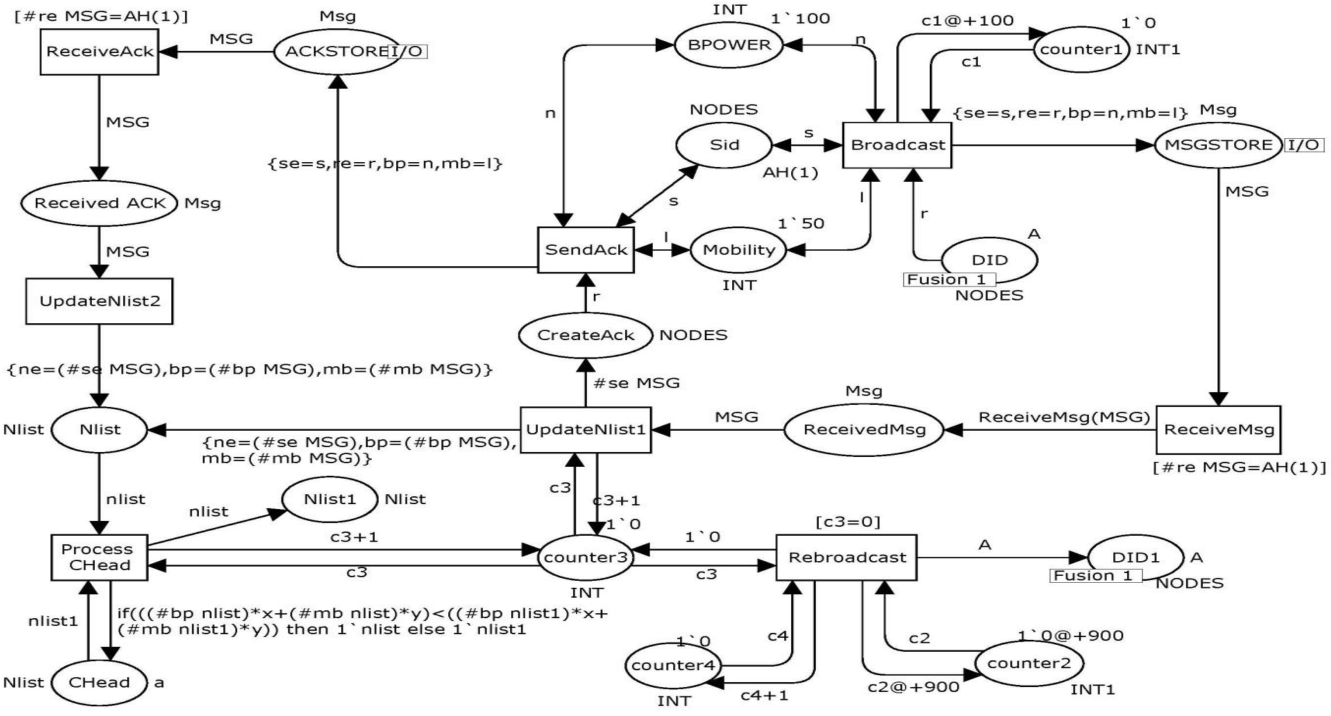

Figure.5 CPN Model of an On-Demand WCA

NODE GENERATING BROADCAST MESSAGES

If a node wants no identify its neighbors list it will broadcast the message to all other nodes. These messages are stored in the MsgStore. The broadcast message contains the ID of the sender and receiver and battery power and mobility of the sender. The message is in the form of a record like {se, re, bp, mb}.

RECEIVING MESSAGE FROM MSGSTORE

If there is any message in the message store for a node it will generate a receiving function, Based on this function the node may or may not receive the message. If the node receives a message then it will update his node list and send back an acknowledgment to the sending node. All the acknowledgements are stored in the AckStore. The Acknowledgement is also in the form of a record like {se, re, bp, mb}, but contain the receiver node values.

RECEVING ACKNOWLEDGEMENT AND UPDATE NODE LIST

If a node has an Acknowledgement in the AckStore with hits address in the receiver field then it will receive the acknowledgement and update his neighbor list. Every node is maintaining its own neighbors list. This will be updated by two transitions, one if it will receive a message from any other node and the second if it will receive an acknowledgement from any other node.

IDENTIFYING THE CLUSTERHEAD

After successfully identifying the neighbors of a node, calculate the Weight for each and every node. Based on these weights among the neighbors which one is having the minimum weight is elected as the cluster head. This will be stored in a place called CHead in the form of a record contain all the details of the cluster head.

DOMINANT SET UPDATES

If a node is unable to identify any neighbors or it is not in the transmission range of any other then it will again rebroadcast messages to every other node. This is called as the dominant set update. For knowing these four counters are required, one is calculating the number of updates for neighbors list, two are calculating the model time and the remaining one is calculating the number of dominant set updates.

C. GLOBAL DECLARATIONS

The global declarations comprise the color sets and the variables for the On-Demand WCA. The color sets show data structures storing different information such as message, acknowledgment, cluster head, neighbors list. The color set Msg defines the both message and acknowledgment. It contains the following fields:

Se: Sender ID of the Message/Acknowledgment.
Re: Receiver ID of the Message/Acknowledgment.
Bp: Battery power of the node.
Mb: Mobility of the node.

The color set Nlist defines the both neighbor’s list and cluster head. It contains the following fields.

Ne: Neighbor/cluster head node ID.
Bp: Battery power of the neighbor/cluster head node.
Mb: Mobility of the neighbor/cluster head node.

The color set NODES contains the all IDs of the nodes.

```
val NoOfNodes=10;
color INT=int;
color INT1=int timed;
color NODES=index AH with
1..NoOfNodes;
color Msg=record
se:NODES*re:NODES*bp:INT*mb:INT;
color Nlist=record
ne:NODES*bp:INT*mb:INT;
var MSG:Msg;
var nlist,nlist1:Nlist;
var s,r:NODES;
var n,l,c1,c2,c3,c4:INT;
fun ReceiveMsg(MSG)=if
discrete(0,100)<2 then 1`MSG else empty;
```

Figure.6 Global Declarations for CPN Model

IV. SIMULATIONS AND VALIDATION EXPERIMENT

For the illustrative purpose, consider a MANET example consisting of 10 nodes. Thus when a node send a broadcast message, it will send 9 copies of the message to the place called MsgStore. In this simulation example taken the average degree of the MANET graph, i.e., the average number of neighbors of a node differently and drawn the graphs for all those values. Thus the probability bar PB can be calculated as the following

$$PB = \frac{d}{(n-1)(n-1)} = \frac{2}{(10-1)(10-1)} = \frac{2}{81} = 0.024$$

Or 100PB = 100 X 0.024 = 2.

Consider 100PB to be 2, an integer, for convenience using in CPN Tool. The other values for PB are also calculated by using different number of d values.

TABLE.1 SIMULATION REPORT FOR THE PB=2

NODE ID	Neighbor Nodes	Cluster head
1	6	1
2	4, 7	4
3	10	10
4	2, 9	4
5	9	9
6	1, 7	1
7	2, 6, 8	2
8	7	8
9	4, 5, 10	4
10	3, 9	9

TABLE.2 SIMULATION REPORT FOR THE PB=10

NODE ID	Neighbor Nodes	Cluster head
1	2	2
2	1, 3, 8	2
3	2, 7	2
4	6, 7, 8	4
5	7	7
6	4	4
7	3, 4, 5, 8, 10	4
8	2, 4, 7, 9	4
9	8	8
10	7	7

For each and every time the TA mechanism gives the different structure for a MANET graph without knowing the actual topology of the MANET [6]. To measure the performance of system we identify three metrics: (i) The number of cluster heads (ii) The number of dominant set updates (iii) The number of reaffiliations. These parameters are studied for varying amount of transmission range (PB in CPN model).

Figure.7 shows the variation of the average number of cluster heads with respect to the transmission range. From this figure observe that the average number of cluster heads decreases with the increase in the transmission range. This is due to the fact that, a cluster head with a large transmission range will cover a larger area.

Figure.7Average. No of Cluster heads Vs PB

Figure.8 shows the number of dominant set updates with respect to the transmission range. For small transmission range, the cluster area is small and the probability of a node moving out of its cluster is high. As the transmission range increases, the number of dominant set updates decreases because the nodes stay within their cluster in spite of their movements.

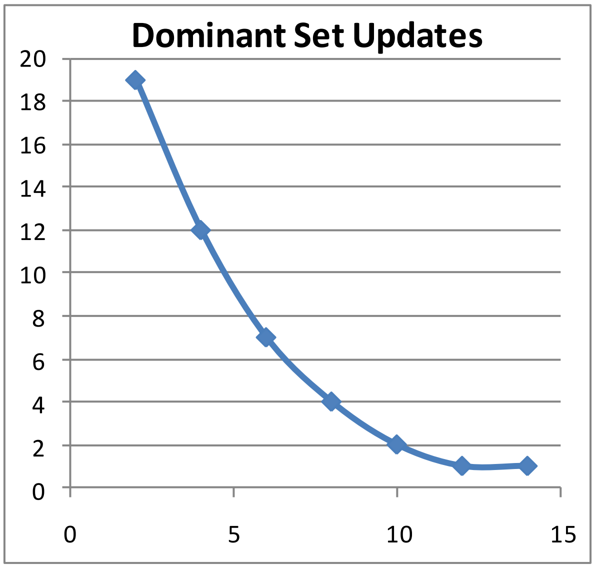

Figure.8 Dominant set updates Vs PB

V. CONCLUSION AND FUTURE SCOPE

There are few formal methods available presently for designing and testing protocols for MANET's due to their dynamically changing graph structures. The simulation results show that the TA mechanism can simulate the mobility of a MANET without knowing its actual topology. On-demand weighted clustering algorithm (WCA) which can dynamically adapt itself with the ever changing topology of mobile ad hoc networks. The WCA has the flexibility of assigning different weights and takes into an account a combined effect of the ideal degree, transmission power, and mobility and battery power of the nodes. The algorithm is executed only when there is a need, i.e., when a node is no longer able to attach itself to any of the existing cluster heads. For the future work plan to animate the different concurrent activities in the model, drawing of state space graph, for the CPN models by using Graphviz tool. And plan to calculate the reaffiliation count and try to evaluate the performance regarding reaffiliations Vs Transmission range.

REFERENCES

[1] *Mobile Ad Hoc Networking* by Stefano Basagni, Marco Conti, Silvia Giordano, Ivan Stojmenovic. IEEE press, A John Wiley & Sons, Inc., Publication (2004).

[2] Chatterjee, M., Das, S. K., and Turgut, D. 2002. *WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. Cluster Computing* 5, 2 (Apr. 2002), 193-204.

[3] K. Jensen *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use – Volume 1: Basic*

Concepts EATCS Monographs in Computer Science, Vol. 26, Springer-Verlag (1992).

[4] K. Jensen *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use – Volume 2: Analysis Methods EATCS Monographs on Theoretical Computer Science*, Springer-Verlag (1994).

[5] CPN Tools. www.daimi.au.dk/CPNtools.

[6] Xiong, C., Murata, T., and Tsai, J. 2002. *Modeling and simulation of routing protocol for mobile ad hoc networks using colored Petri nets*. In Proceedings of the Conference on Application and theory of Petri Nets: Formal Methods in Software Engineering and Defense Systems - *Volume 12* (Adelaide, Australia). C. Lakos, R. Esser, L. M. Kristensen, and J. Billington, Eds. ACM International Conference Proceeding Series, vol. 145. Australian Computer Society, Darlinghurst, Australia, 145-153.*Application of Petri nets to communication networks: advances in Petri nets*/ Jonathan Billington (Ed.). Springer, 1999.

[7] Jensen, K., Kristensen, L. M., and Wells, L. 2007. *Colored Petri Nets and CPN Tools for modeling and validation of concurrent systems*. Int. J. Softw. Tools Technol. Transf. 9, 3 (May. 2007), 213-254.

[8] Tadao Murata, *Petri Nets: Properties, Analysis and Applications*, in Proceedings of the IEEE, Vol. 77, No. 4, pp.541-580, April 1989.

[9] *Application of Petri nets to communication networks: advances in Petri nets*/ Jonathan Billington (Ed.). Springer, 1999.

APPENDIX

DECLARATIONS FOR CPN MODEL

```
12. val x=2;
13. val y=3;
14. val A=1`AH(2)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(6)++1`AH(7)++
       1`AH(8)++1`AH(9)++1`AH(10);
15. val B=1`AH(1)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(6)++1`AH(7)++
       1`AH(8)++1`AH(9)++1`AH(10);
16. val C=1`AH(1)++1`AH(2)++1`AH(4)++1`AH(5)++1`AH(6)++1`AH(7)++
       1`AH(8)++1`AH(9)++1`AH(10);
17. val D=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(5)++1`AH(6)++1`AH(7)++
       1`AH(8)++1`AH(9)++1`AH(10);
18. val E=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(4)++1`AH(6)++1`AH(7)++
       1`AH(8)++1`AH(9)++1`AH(10);
19.    val F=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(7)++
       1`AH(8)++1`AH(9)++1`AH(10);
20. val G=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(6)++
       1`AH(8)++1`AH(9)++1`AH(10);
21. val H=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(6)++
       1`AH(7)++1`AH(9)++1`AH(10);
22. val I=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(6)++
       1`AH(7)++1`AH(8)++1`AH(10);
23. val J=1`AH(1)++1`AH(2)++1`AH(3)++1`AH(4)++1`AH(5)++1`AH(6)++
       1`AH(7)++1`AH(8)++1`AH(9);
24. val a={ne=AH(1),bp=100,mb=50};
25. val b={ne=AH(2),bp=80,mb=60};
26. val c={ne=AH(3),bp=120,mb=80};
27. val d={ne=AH(4),bp=70,mb=50};
28. val e={ne=AH(5),bp=150,mb=40};
29. val f={ne=AH(6),bp=120,mb=60};
30. val g={ne=AH(7),bp=90,mb=70};
31. val h={ne=AH(8),bp=80,mb=65};
32. val i={ne=AH(9),bp=75,mb=85};
33. val j={ne=AH(10),bp=140,mb=60};
```