Modeling of a Topology Adaptive Clustering Algorithm for Mobile Ad Hoc Networks using Coloured Petri Nets

Suchismita Chinara* and Santanu Kumar Rath**

***Abstract*-** The modeling of a communication protocol for a distributed computing system requires making clear inputs and outputs of the remote communicating entities and the formal modeling of the communication channel in the aim of protocol verification or validation. Petri nets are the formal tool which is mostly used for the purpose of modeling and validation. The coloured Petri Net model of a system can be an executable model representing the different states of the system as well as the events that can cause the system to change its states. In this paper we model a distributed topology adaptive clustering algorithm for mobile ad hoc networks by using the coloured petri net tools.

Key Words: Clustering, Node Weight, Transition, Place.

I. Introduction

The nodes of the mobile ad hoc network (MANET) are equipped with wireless transceivers which move freely while remaining reachable to each other. With limited transmission range the nodes are capable to communicate with each other using intermediate relays or multi-hop wireless links. So the basic assumption for the mobile ad hoc network is that a node can play the role of a router for forwarding the packets of its neighbors [1]. The node movement in the dynamic environment causes frequent topology changes disturbing the path of packet transmission. Further the scarcity of radio resources and bandwidth, limited battery power and computing power of nodes pose challenge in MANET scalability and efficiency [2]. Such a scenario demands for the partitioning of the network into logical groups so that the above challenges could be met efficiently.

*Suchismita Chinara is with the Department of Computer Science Engineering, National Institute of Technology, Rourkela -769008, India (Phone: 91-661-2462361; e-mail: suchismita@nitrkl.ac.in).

**Santanu Kumar Rath is with the Department of Computer Science Engineering, National Institute of Technology, Rourkela -769008, India (e-mail: rath.santanu@gmail.com).

In the cellular architecture, the presence of a fixed infrastructure such as a base station allows the host to communicate directly with itself reducing the wireless part of communication to a single hop problem. Many good solutions have been proposed till now for handling the mobility of hosts by this base station. So the concept of mapping the cellular architecture into a peer to peer network leads to the formation of clusters [3]. Every cluster posses a cluster head that plays the role of a base station in allocating resources to the one-hop member nodes and lies responsible for the inter-cluster communication [4]. The set of cluster heads form the virtual back bone of the ad hoc network for packet transmission.

The organization of the paper is made as follows. Section II describes about the basics of the coloured petri nets followed by the related work done so far in the areas of clustering in MANET and use of CPNet in communication in section III. Section IV describes in detail about the topology adaptive clustering algorithm for mobile ad hoc network followed by Section V that models the algorithm using Coloured Petri Net tools. Finally, section VI concludes the paper.

II. Related Work

Clustering in mobile ad hoc networks is a well studied area where selected nodes act as the cluster heads for their one-hop neighbors. Among the various clustering algorithms, the lowest ID algorithm proposed by Ephremides, Wieselthier and Baker [4] is an ID based algorithm where a node having the lowest identification number among its immediate neighbors is selected as the cluster head. This algorithm retains its utility as a benchmark for its simplicity and producing reasonably stable cluster control architecture as discussed by Gerla and Tsai in [5]-[6].

A node connectivity based algorithm was proposed by Parekh [2] that aims to achieve minimum number of cluster heads in the network. A node with highest degree of node connectivity among its neighbors is

selected as the cluster head and its one hop neighbors are the members of the cluster head. This algorithm results in a smaller dominant set. Thus the faster communication is guaranteed.

The mobility metric based version of lowest ID algorithm MOBIC was proposed by Basu, Khan and Little [7]. The authors use the ratio of two consecutive signal strengths received by a particular node from its neighbors to predict its relative motion with respect to that neighbor. The variance of entire set of relative mobility values of a node with respect to its neighbors provides its relative mobility in the network. Once the relative mobility metric for every node is decided, MOBIC is called upon the nodes which works almost same as the Lowest ID algorithm, where the node IDs are replaced by the relative mobility metrics of each node.

The authors in [8]-[9] proposed a distributed mobility adaptive algorithm DMAC which eliminates the assumption of non-mobility of the mobile hosts during clustering setup and maintenance. Consideration of the local topology of the network for selection of the cluster head is the major strength of this algorithm. Further,The authors of [10] proposed a generalized version of DMAC where K cluster heads are allowed to remain as neighbors and reaffiliation by a node occurs only when the difference in weight values of two neighbor cluster heads exceed a threshold value H.

The combined metric clustering algorithms [11] - [13] use the node parameters like running average, degree of connectivity, transmission power, available battery power to find its suitability as a head. But obtaining so much of information (specially, mean connectivity in a dynamic network) to compute the combined weight for every node in the network itself needs a longer frozen period of motion before the cluster is actually formed. A large number of message exchanges take place globally to yield the node with lowest weight.

All the above clustering algorithms have emphasized on either reducing the number of cluster heads or number of cluster head updations. But none have given emphasis on the energy consumption of the mobile nodes. Similarly, the improvement of network life time has not been considered by any of the articles earlier. In this article a topology adaptive clustering algorithm has been modeled which enhances the network life time by handling the node energy in an efficient manner.

III. The Topology Adaptive Clustering Algorithm

The preliminary version of the clustering algorithm is presented in [14]. Here the mobile ad hoc network can be modeled as a graph $G = (V, L)$, where V is a finite set of mobile nodes and L is a finite set of links that exist between the nodes. In the dynamic network the cardinality of the nodes $|V|$ remains constant, but the cardinality of links $|L|$ changes due to the mobility of the nodes. Each node $v \in V$ is uniquely identified by an integer identifier and has a wireless transmission range v_{trange}. When a node v_1 is within the transmission range of v_2, they are assumed to be connected by a unidirectional link $l_{12} \in L$, such that whenever v_1 broadcasts a message, it is received by v_2 via l_{12}. Similarly, when v_2 is within the transmission range of v_1, they are assumed to be connected by an unidirectional link $l_{21} \in L$, such that whenever v_2 broadcasts a message, it is received by v_1 via l_{21}.

A. Neighbors Detection Protocol (NDP)

In order to design the topology adaptive clustering algorithm, **NDP** has been proposed so that it provides a mechanism to detect the one-hop neighbors of the nodes in the ad hoc network. The protocol works as follows:

Step 1: Node u broadcasts the ***Neighbor Detection Packet*** **(NDPAK)** to its neighbors.
Step 2: The packet is received by node v which is within the transmission range of u. Node v sends back the N***eighbor Acknowledge Packet*** (**NAC**) to$u$ along with its own information enclosed in the packet.
Step 3: Sender u updates its neighbor table (**NTAB**) by adding v as its neighbor and its updated information.
Step 4: u sends back a N***eighbor Confirmation*** (**NC**) packet so that v updates its own neighbor table and a bidirectional link is established between the two nodes.
The data structure for the NDPAK is as:

SID	RID	STR/RTR	NRQ/NAC	WT	STATUS

SID: **Source Identification**
RID: **Receiver Identification**
X: ***ALL***
NUM: ***node with identifier NUM***

STR: **Source Transmission Range**
RTR: **Receiver Transmission Range**
NRQ/NAC:
NRQ: The neighbor request packet
NAC: Neighbor acknowledge packet

WT: **Weight of the packet sender**
STATUS: Node status of the packet sender
0: Uncovered
1: Cluster Member
2: Cluster Head

Here, the NRQ/NAC field distinguishes a neighbor request packet from a neighbor acknowledge packet. The weight of the node specifies its capability to be selected as a cluster head. The STATUS field of the node indicates its current role in the network. It is set to 0 when the network is first initialized as the role is undefined before the cluster is actually formed. Subsequently this field is filled with appropriate value. Fig 1 denotes the example of a network topology with five nodes.

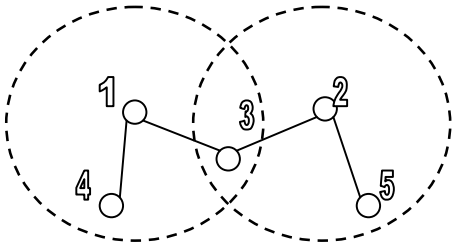

Fig. 1 A sample network topology

Here, the nodes are identified by unique integer numbers from 1 to 5. The transmission ranges for the nodes 1 and 2 are indicated by the dotted circles around it. It is clear from the figure that node 3 lies in the transmission range of both nodes 1 and 2 whereas node 4 and 5 are exclusively in the range of 1 and 2 respectively. The structure of a NDPAK sent by node 1 is as:

1	**X**	1_{trange}	**NRQ**	**WT (1)**	**0**

The receiver field of the packet has the value X to indicate that it is to be received by all the nodes that are in the transmission range. WT (1) value indicates the current weight of the node 1. This packet will be received by both node 3 and 4 and they will send back the NAC packet to acknowledge it. The acknowledge packet (NAC) sent back by node 3 to node 1 is as:

3	**1**	3_{trange}	**NAC**	**WT(3)**	**0**

For the topology of figure 1, where node 3 receives the NRQ packet from both nodes 1 and 2, it sends back the NAC packets accordingly. The arrival of the NAC packets informs the sender about the presence of its one-hop neighbors and it updates its own neighbor table **NTAB** accordingly. The data structure for the NTAB is as:

NID	**DIST**	**NS**	**NTR**	**NWT**

NID: ***Neighbor ID***
DIST: ***Neighbor Distance***
NS: ***Neighbor Status***
NTR: ***Neighbor Transmission Range***
NWT: ***Neighbor Weight***

The neighbor distance DIST can be calculated from the signal strength of the received signal from the neighbor. However, for the current work the Euclidian distance between two nodes is considered as the actual distance between them. When a NAC packet arrives, the source node updates its NTAB accordingly and sends a NC packet to confirm its link with the later so that a bidirectional link is established between both of them.

B. Calculation of the node weight

The mobility of nodes changes the network topology frequently which in turn hampers the network stability. So choosing the nodes with minimum mobility to form the virtual back bone is preferred. This ensures better backbone stability. Similarly the limited battery power devices consume their energy and become dead while routing the packets through them. This delinks the path for packet routing and demands for further establishment of routing back bone. In order to ensure the availability of routers in the routing backbone, nodes with more available battery power are chosen as the back bone forming nodes. Keeping these factors in mind, the node weights are calculated by considering the node mobility and its available battery power as the key values. Here δ is assumed to be the maximum permissible speed of a node in the network. Thus the mobility factor of every node is calculated by computing the difference of δ and its average speed during a certain time interval. A larger mobility factor

indicates a node with less mobility and vice versa. The available battery power is the energy associated with the node at the instant of weight calculation. These two parameters are added with different weight factors to find the individual node weights. The steps for calculating the weights are described below:

Step 1: The total distance covered by a node during last n time units is

$D_v = \sum_{i=t-n}^{i=t} dist_v$ where t= the current time.

Compute average speed of a node

$Sv = D_v / n.$

Step 2: Compute Mobility factor $\Delta M = \delta - Sv$. i. e. How far is the average speed of the node from δ.

Step 3: Compute available battery power as

$P_{av} = P_{av} - P_{cons}$ where

P_{av} = Currently available battery power of the node (Initially it is the maximum battery power).

P_{cons} = Battery power consumed by the node from time to time.

Step 4: Compute the weight of the node as

$WT(v) = x_1 \Delta M + x_2 P_{av}$

Where x_1 and x_2 are the weight factors that are normalized so that

$x_1 + x_2 = 1.$

D. Selection of Volunteer Cluster Head

After the weight calculation of the nodes, the following algorithm is called upon to select the set of **volunteer cluster heads**.

PSEUDO CODE FOR VOLUNTEER CLUSTER HEAD SELECTION

For (every $v \in V$)
If $wt_v \rangle wt_i$ where $i \in \Gamma(v)$ // $\Gamma(v)$ is the neighbor set of v
Then Set head= v
For (every $x \in \Gamma(v)$)
If STATUS (x) = 0 Then
Set HEAD (x) = head
End for
End for

The algorithm indicates that a node having maximum weight among its 1-hop neighbors declares itself as the volunteer cluster head. And its 1-hop uncovered neighbors (i.e. whose role is not yet decided) become the members of the volunteer head. The set of covered nodes are exempted from taking part in subsequent selection procedure and this process is repeated till all the nodes are assigned with their role either as a cluster head or a cluster member.

During the cluster head selection phase every node broadcasts it's ID along with its weight Wt_i to all neighbors and stores the weights Wt_j that it hears from other nodes. If it does not hear another node ID with weight higher than itself then it declares itself as a volunteer cluster head and its one-hop uncovered neighbor nodes become its members. In case of a tie in the node weights the low ID node is preferred for the role of cluster head. Unlike [9] once a member node is affiliated with a cluster head, it does not re-affiliate to a new head unless it goes out of the range of its current head or the head drains out of battery power. This reduces the number of re-affiliations lowering the cluster maintenance overhead.

The example of a clustered topology as the result of the above algorithm is demonstrated with the help of Fig 2. Here every node is identified with a unique ID and its associated weight in parenthesis. We assume that the weights are already being computed for every node. The link between every pair of nodes denotes that they are within the transmission range of each other and establish a bidirectional link among them. Volunteer cluster heads are identified with dark circled nodes after the exchange of their weights within the local topology. The set of cluster heads changes with the change in topology of the dynamic network.

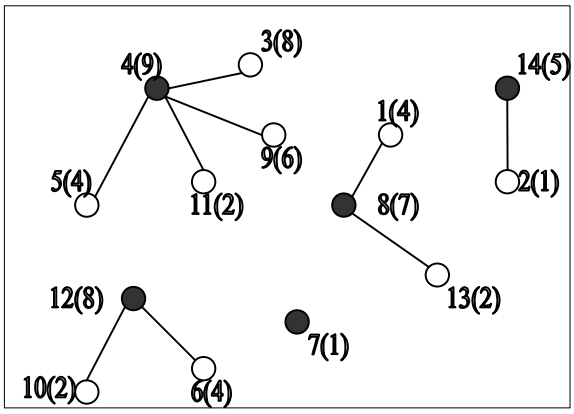

Fig. 2 Volunteer cluster heads are elected to form clusters

Every node in the network maintains its own cluster table. The structure of the cluster table (CTAB) is as:

MID	MWT	MDIST	MT

MID : Member ID
MWT : Member Weight
MDIST : Member Distance
MT : Member Transmission Range

The cluster table is updated as the node becomes a head node and with the occurrence of node reaffiliation and reelection. That is when a member of the current head goes out of its transmission range it is deleted from the CTAB. And at the same time new row is added when a new node becomes the member of this cluster head. In figure 2, cluster head 12 has two members where as head 7 is an isolated head without having any member affiliated to it.

IV. Modelling Of The Algorithm Using Coloured Petri Net

A. Coloured Petri Nets

Coloured Petri Nets (CPNs) is a discrete event modeling language that combines Petri Nets with the standard modeling language (ML) that provides the primitives for the definition of data types, describing data manipulation and for creating compact and parameterisable models [15]. CPN is a graphical language for constructing models of communication protocols, data networks, distributed algorithms and embedded systems. It is also applicable for modeling system where concurrency and communications are the key characteristics. CPNs can be simulated interactively (single-step debugging method) or automatically (similar to program execution).

In contrast to most specific languages, Petri nets are state and action oriented at the same time. That is the modeler can determine freely whether at a given instance of time he desires to concentrate on states or actions [16]-[18]. Following are few CP Net definitions that are used in the paper for the modeling purpose:

Places: The circle or the oval of the graphical representation that indicates the states of the modeled system. The name of the place is written inside the circle. Every place has one or more resources available with it. They are called tokens.

Transitions: The rectangular boxes that indicate the actions that could occur when some preconditions are satisfied, are called transitions. The firing of a transition could remove / add a token from / to a place. The name of the transition is written inside the rectangle.

Arc: The directed arrow that joins a place to a transition is called an arc. An arc could be incoming or outgoing.

Token Color: Every place has a color set that tells the type of the data it can handle. This is called the token color. The types are similar to the types in programming language.

Marking: The state of a CP-Net is called the marking. It is the number of tokens positioned on the individual places. The tokens that are present on a particular place are called the marking of that place.

B. Modeling of TACA using CPNets

To prove the correctness of the proposed clustering algorithm, a formal model has been created and analyzed to check if it could provide the required output. Coloured Petri Nets have the capability to handle the non-determinism, concurrency and different level of abstraction. Fig 3 shows the top level of the nodes where 10 nodes are considered for the modeling with two places ACKSTORE and MSGSTORE that stores all acknowledgements and broadcast messages respectively. The bidirectional links indicate the two way communication between the nodes and the message stores. The function of the places MSGSTORE and ACKSTORE are as follows:

MSGSTORE: The sender node broadcasts its own parameters to the other nodes and they are stored in the place MSGSTORE after the firing of the transition Broadcast. Every node checks the place MSGSTORE for the broadcasted messages destined for it. When it finds a message present in the MSGSTORE, it retrieves the message by generating a receiving function.

ACKSTORE: This place holds the acknowledgement messages sent by the nodes. As a node sends any acknowledgement, it gets stored in this place. The other nodes retrieve the acknowledgements destined for them from this place.

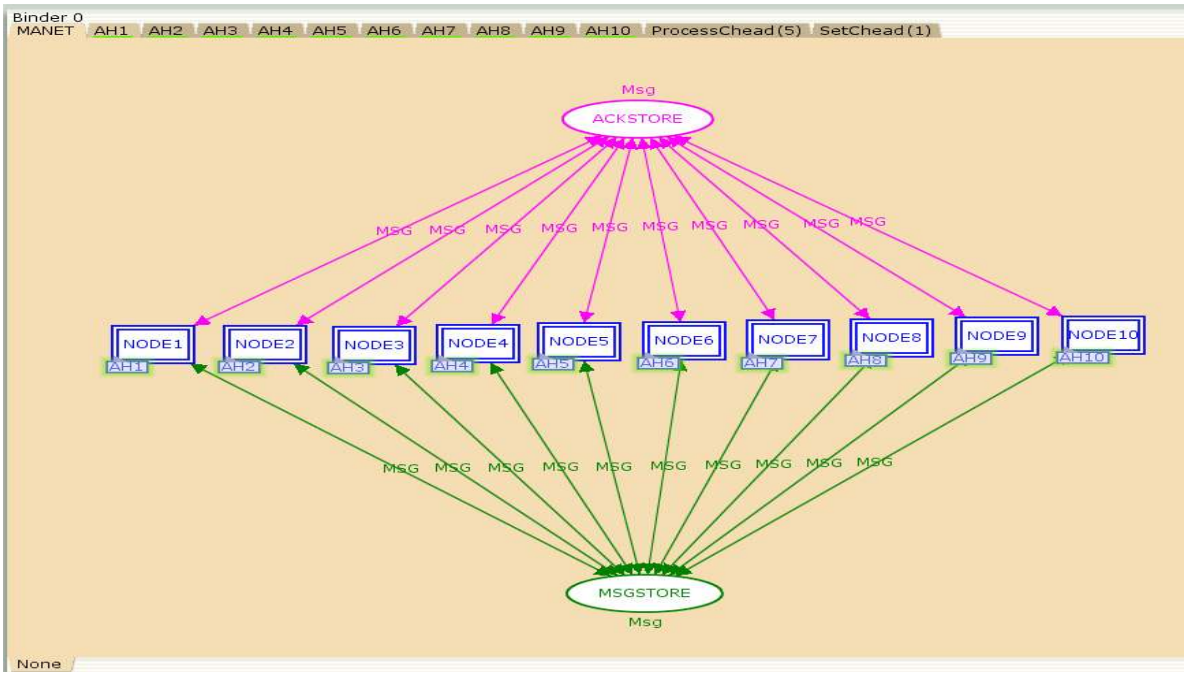

Fig. 3 Top level of the model.

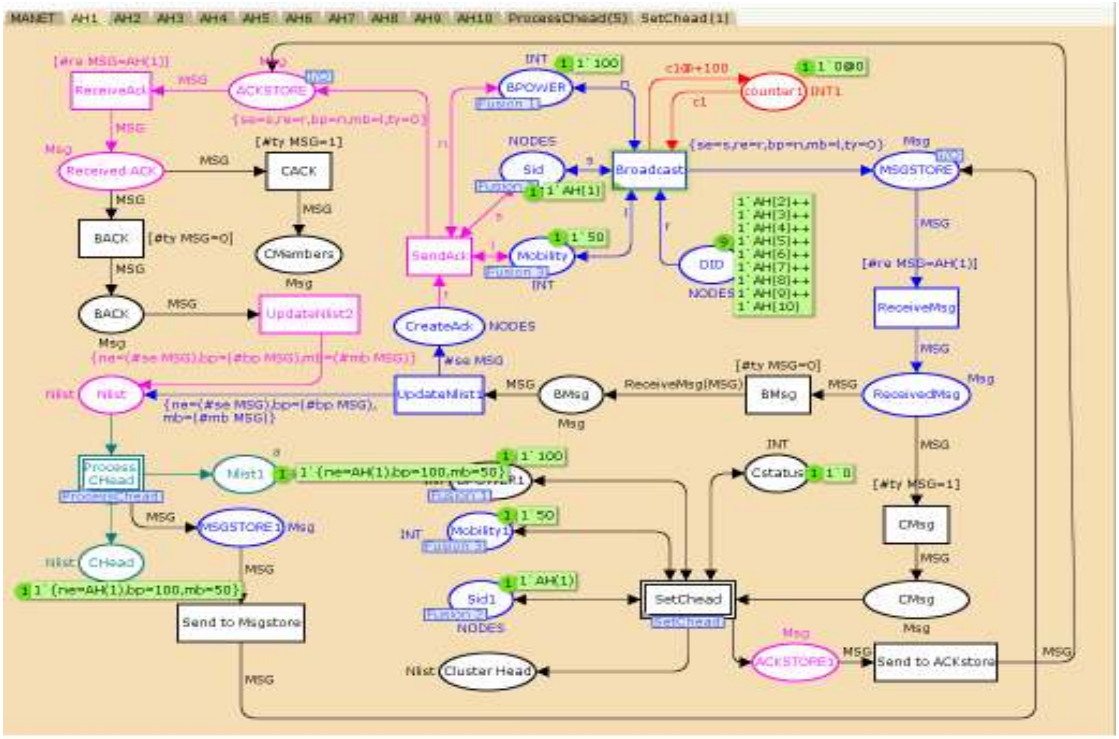

Fig. 4 Detailed model of a single node.

The detailed model of a single node in the mobile network is shown in Fig 4. The node parameters like battery power, source identification and node mobility are represented by the places BPOWER, Sid and Mobility respectively. When the transition Broadcast is fired, the concern node parameters of the sender node are broadcasted into the network. The list of the receiving nodes which may be able to receive the message (if within the transmission range of the sender node) are listed in the destination identification place DID. The message fired by the transition is stored in the place MSGSTORE so that other nodes receive it through their receiving function. The output of the above transition is the message denoted as MSG which is Boolean type with value 0 or 1. When the type of MSG is 0, it is considered as a broadcast message from a non-cluster head node whereas if the MSG type is 1, it is considered to be a broadcast message from a cluster head. The broadcast message received from a non-cluster head node helps only to update the neighbor list of the receiving node as the transition UpdateNlist1 is fired. The transition UpdateNlist1 has two possible outcomes. First, it receives the message with all the node parameters so that the neighbor list of the receiving node is updated. Second, it creates an acknowledgement packet for the source node and is sent to the place ACKSTORE after the firing of the transition SendAck. The transition SendAck also gets the input from the places Sid, BPOWER and Mobility to create the acknowledge message.

As the nodes update their neighbor list by the exchange of messages between them, the weight is calculated for every node considering the node parameters. The module Process CHead calculates the weights of all the nodes and stores the node with maximum weight in the place CHead. This selected node is then declared as the volunteer cluster head among its own neighbors.

Fig 5 depicts the detail model of the Process CHead that indicates the process how a cluster head is selected. After the head selection is over, it sends a broadcast message to other nodes indicating its status as the head node. This is done with the firing of the transition CBMSG that stores the message in the MSGSTORE place where the message type is set to be 1 as discussed earlier. The module SetChead sends messages to all its neighbors telling them that it is their cluster head and they need to register them with it. Then it sends back acknowledgement to all other nodes. The neighbors who receive the message from the cluster head update their neighbor list and the cluster table CTAB accordingly.

VI. Conclusion

Clustering is a proven solution in mobile ad hoc networks to meet the challenges of network scalability, battery power etc. The current article models a topology adaptive clustering algorithm using the Colour Petri Net tools. The model depicts the detail flow of the inputs and outputs that takes place for the cluster formation in the ad hoc network. The individual modules are described to show the details of the model. The model can help the network designer in providing a road map for the future simulation.

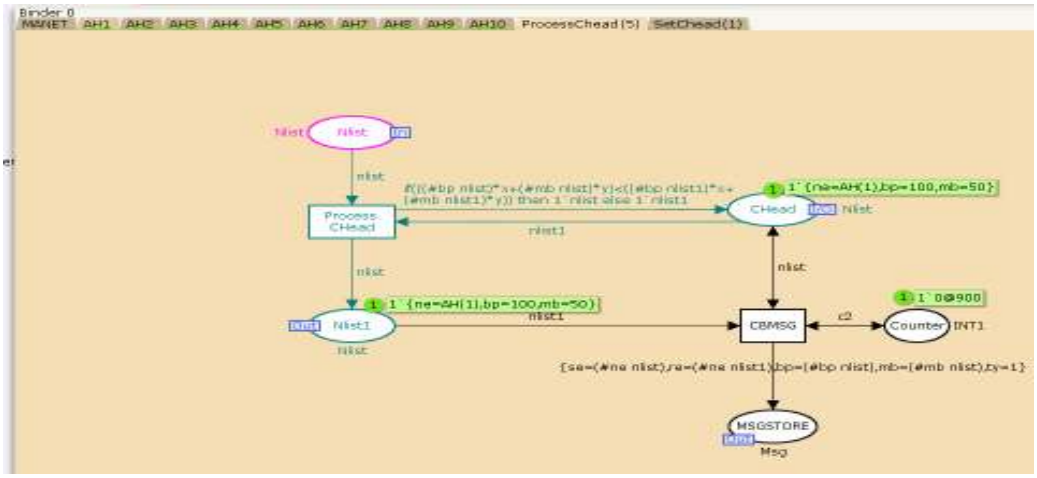

Fig. 5 The detail model of the module Process Chead.

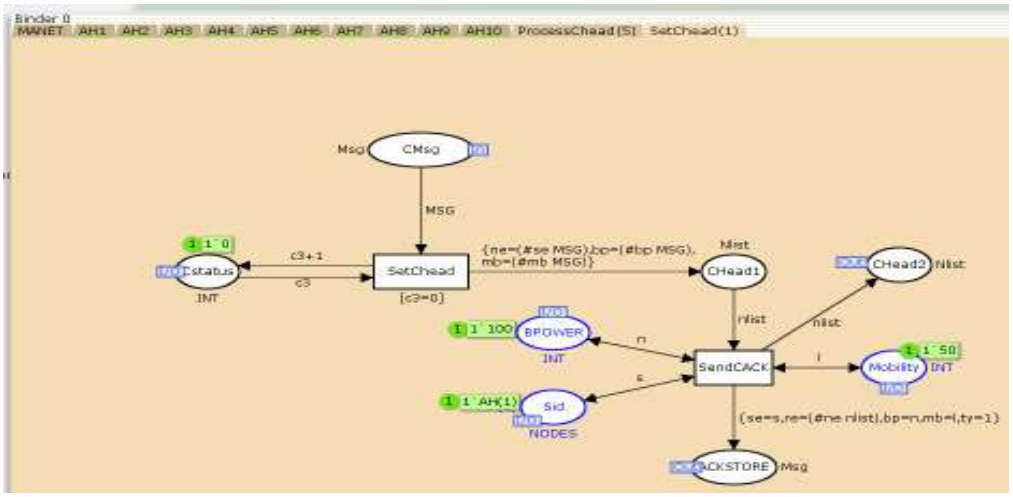

Fig. 6 The detail model of the module SetChead.

REFERENCES

1. S.Basagni, M. Conti, S.Giordano and I.Stojmenovic, Mobile ad hoc networking, IEEE Press, Wiley Interscience, New Jersey, 2004.
2. A.K.Parekh, Selecting routers in ad hoc wireless network, in: Proc. of the SBT/IEEE International Telecommunication Symposium, ITS, August 1994.
3. S. Basagni, I. Chlamtac, A. Farago, A generalized clustering algorithm for peer-to-peer networks, in: Proc. of Workshop on Algorithmic aspect of communication (satellite workshop of ICALP), July 1997.
4. D.J.Baker, J.E.Wieselthier and A. Ephremides, A design concept for reliable mobile radio networks with frequency hopping signaling, in: proc. of IEEE, 75 (1) (1987) 56-73.
5. M.Gerla and J.T.C.Tsai, Multicluster, mobile, multimedia radio network, Journal of Wireless Networks, vol.1, no.3 (1995), 255-265.
6. S. Chinara and S. K. Rath, A survey on One-Hop Clustering Algorithms in Mobile Ad Hoc Networks, Journal of Networks System Management, Springer Science, 17 (2009), 183-207.
7. P.Basu, N.Khan and T.D.C. Little, A mobility based metric for clustering in mobile ad hoc networks, in: Proc. of IEEE ICDCS 2001 Workshop on Wireless Networks and Mobile Computing, Phoenix, AZ, 2001.

8. S.Basagni, Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks, in: Proc. of Vehicular Technology Conference, VTC, vol.2, 1999, pp. 889-893.
9. S. Basagni, Distributed clustering for ad hoc networks, in: Proc. of International Symposium on Parallel Architectures, Algorithms and Networks, June 1999, pp. 310–315.
10. R. Ghosh and S. Basagni, Limiting the impact of mobility on ad hoc clustering, in: Proc. of the 2nd ACM International Workshop PE-WASUN '05, Montreal, CA, 2005, pp. 197--204.
11. S.Das, M.Chatterjee, D.Turgut, Wca: A weighted clustering algorithm for mobile ad hoc networks, Journal of Cluster Computing (special issue on mobile ad hoc networks), vol. 5, (2) (2002), 193-204.
12. S.Das, M.Chatterjee, D.Turgut, An on demand weighted clustering algorithm (WCA) for ad hoc networks, in: Proc. of IEEE GLOBECOM 2000, San Francisco, 2000, pp. 1697-1701.
13. Wei-dong Yang and G-z Zhang, A weight-Based Clustering Algorithm for mobile Ad Hoc Network, in: Proc. of Third International Conference on Wireless and Mobile Communication (ICWMC'07), Guadeloupe, March 2007.
14. S. Chinara and S.K. Rath, TACA: Topology Adaptive Clustering Algorithm for Mobile Ad Hoc Networks, in: Proc. of International Conference on Wireless Networks ICWN09, Las Vegas, vol.2, July 2009, pp. 391-397.
15. K. Jensen, L. M. Kristensen and Lisa Wells, "Coloured Petri Nets and CPN Tools for modeling and validation of concurrent systems", International journal on Software Tools Technol Transfer, pp. 213- 254, 2007.
16. K. Jensen, An introduction to the practical use of Coloured Petri Nets, Lecturer Notes in Computer Science, vol. 1492, 1998, pp. 237-292.
17. W. Reisig, Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets, Springer, Berlin (1998).
18. L.M. Kristensen, S. Christensen and K. Jensen, "The Practitioner's Guide to Coloured Petri Nets", International Journal on Software Tools for Technology Transfer, vol. 2, 1998, Springer-Berlin, pp.98-132.