Neighborhood Detection in Mobile Ad-Hoc Network Using Colored Petri Net

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

Bibhuti Bhusan Panigrahi(10506045)

&

Sunil Kumar Parida(10506025)

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2009

Neighborhood Detection in Mobile Ad-Hoc Network Using Colored Petri Net

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

Bibhuti Bhusan Panigrahi(10506045)

Sunil Kumar Parida(10506025)

Under the Guidance Of

Prof. S. K. Rath

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2009

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, “NEIGHBORHOOD DETECTION IN A MOBILE AD-HOC NETWORK USING COLORED PETRI NET” submitted by Sri Bibhuti Bhusan Panigrahi and Sri Sunil Kumar Parida in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. S.K. Rath
Dept .of Computer Science and Engineering
National Institute of Technology
Rourkela – 769008

Acknowledgement

We express our sincere gratitude to Prof. S.K. Rath, Department of Computer Science and Engineering, National Institute of Technology, Rourkela, for his valuable guidance and timely suggestions during the entire duration of our project work, without which this work would not have been possible.

We would also like to convey our deep regards to all other faculty members and staff of Department of Computer Science and Engineering, NIT Rourkela, who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on our part to finish this project work.

Date:

Bibhuti Bhusan Panigrahi
Roll No: 10506045

Date:

Sunil Kumar Parida
Roll No: 10506025

Contents

Abstract

Colored Petri Nets (CPNs) [2] is a language for the modeling and validation of systems in which concurrency, communication [6], and synchronization play a major role. Colored Petri Nets is a discrete-event modeling language combining Petri nets with the functional programming language Standard ML. Petri nets provide the foundation of the graphical notation and the basic primitives for modeling concurrency, communication, and synchronization. Standard ML provides the primitives for the definition of data types, describing data manipulation, and for creating compact and parameterizable models. A CPN model of a system is an executable model representing the states of the system and the events (transitions) that can cause the system to change state [4]. The CPN language makes it possible to organize a model as a set of modules, and it includes a time concept for representing the time taken to execute events in the modeled system.

In a mobile ad-hoc network(MANET) mobile nodes directly send messages to each other via wireless transmission. A node can send a message to another node beyond its transmission range by using other nodes as relay points, and thus a node can function as a router [1]. Typical applications of MANETS include defense systems such as battlefield survivability and disaster recovery. The research on MANETs originates from part of the Advanced Research Projects Agency(ARPA) project in the 1970s [1]. With the explosive growth of the Internet and mobile communication networks, challenging requirements have been introduced into MANETs and designing routing protocols has become more complex. One approach for ensuring correctness of an existing routing protocol is to create a formal model for the protocol and analyze the model to determine if indeed the protocol provides the defined service correctly. Colored Petri Nets are a suitable modeling language for this purpose as it can conveniently express non-determinism, concurrency and different levels of abstraction that are inherent in routing protocols.

However, it is not easy to build a CPN model of a MANET because a node can move in and out of its transmission range and thus the MANET's topology dynamically changes. In this paper we propose an algorithm for addressing such mobility problem of a MANET [1]. Using this algorithm a node can find its neighbors ,which are dynamically changing, at any instant of time .

Keywords: Colored Petri Nets, Mobile ad-hoc Network(MANET), Behavioral Modeling, CPN Model, Simulation, Verification.

List Of Figures

List of Abbreviations

CPN	Colored Petri net
ML	Modeling Language
MANET	Mobile Ad-hoc Network
ARPA	Advanced Research Projects Agency
Pr/T NET	Predicate Transition Net
TA	Topology Approximation
PB	Probability Bar

Chapter-1

INTRODUCTION

Introduction:

CPnets is a discrete-event modeling language combining Petri nets [2] and the functional programming language CPN ML which is based on Standard ML [4]. The CPN modeling language is a general purpose modeling language, i.e., it is not focused on modeling a specific class of systems, but aimed towards a very broad class of systems that can be characterized as concurrent systems. Typical application domains of CP-nets are communication protocols [6], data networks , distributed algorithms , and embedded systems. CP-nets are, however, also applicable more generally for modeling systems where concurrency and communication are key characteristics.

A CPN model [4] of a system describes the states of the system and the events (transitions) that can cause the system to change state. By making simulations of the CPN model, it is possible to investigate different scenarios and explore the behaviors of the system. Very often, the goal of simulation is to debug and investigate the system design. CP-nets can be simulated interactively or automatically. An interactive simulation is similar to single-step debugging [2]. It provides a way to “walk through” a CPN model, investigating different scenarios in detail and checking whether the model works as expected. During an interactive simulation, the modeler is in charge and determines the next step by selecting between the enabled events in the current state [5]. It is possible to observe the effects of the individual steps directly on the graphical representation of the CPN model. Automatic simulation is similar to program execution. The purpose is to simulate the model as fast as possible and it is typically used for testing and performance analysis. For testing purposes, the modeler typically sets up appropriate breakpoints and stop criteria. For performance analysis the model is instrumented with data collectors to collect data concerning the performance of the system [5].

Mobile ad hoc networking is a technology for the cooperative management of a group of mobile nodes without requiring established infrastructure and centralized administration. In a MANET mobile nodes directly send messages to each other via wireless transmission [1]. A node can send a message to a destination node beyond its transmission range by using other nodes as relay points and thus a node can function

as a router. This mode of communication is known as *wireless multihopping*. Compared with its hardwired counterpart, a MANET has additional constraints such as bandwidth-constrained, energy-constrained and limited physical security. Because of these additional constraints and the dynamic topology, conventional routing protocols are not appropriate for MANET's. Many routing protocols such as Ad Hoc On-Demand Distance Vector Routing(AODV), Dynamic Source Routing(DSR) [1] etc. that are specific to mobile networking have been proposed. To ensure correctness of a routing protocol, first a formal model for the routing protocol is created and then the model is analyzed to determine if indeed the protocol provides the defined service correctly. CPNs are a suitable modeling language for verification task as they can conveniently express non-determinism, concurrency and different levels of abstraction that are inherent in routing protocols [6]. A major benefit of using CPNs is to obtain complete and unambiguous specifications in the design stage of a routing protocol, and verify if the routing protocol indeed provides the defined service correctly. Design/CPN is a suite of tools for editing, simulating and analyzing CPNs and it has a graphical editor that allows the user to create and layout different net components [4]. However, it is not easy to build a CPN model of a MANET because nodes(hosts) can move in and out of their transmission ranges and thus MANET's topology(graph) changes dynamically.

Currently, there are few mechanisms proposed to model a system's dynamic structure using CPN. Findlow and Billington [8] proposed a method to build a CPN model for a dynamic dining philosophers system whose structure changes in a predictable way [1]. The method uses tokens in a place to describe neighboring pair of philosophers to record the system structure of philosophers' relevant position, and the system's dynamic structure is described by changes of tokens. If a new philosopher A joins the table and sits between philosophers B and C, the system's structure will definitely change in this way: two tokens representing new neighboring information (B,A) and (A,C) are added to the place and one old token (B,C) is removed from the place. If a philosopher A who sits between philosophers B and C leaves the table, the structure will change in this way: two corresponding tokens (B,A) and (A,C) are removed from the place and one token (B,C) is added to the place. Though the proposed method solves the problem of modeling dynamic structure of this system, it is

difficult to capture structure changes of a MANET because the structure of a MANET changes in an unpredictable way. Nodes in a MANET move at will and their movements change neighboring relationship unpredictably [1]. Thus, in this paper we propose an algorithm to address the problem of mobility(dynamic changing topology) in MANETs. By using this algorithm a node can detect its neighbors, which are dynamically changing, at any particular instant of time.

In the next chapter, we have discussed the basic concepts of Petri nets and also different types of Petri nets and why colored Petri net is a suitable tool for implementing the algorithms.

Chapter-2

PETRI NET CONCEPTS

- Introduction
- Original Petri Nets
- Weighted Petri Nets
- Other Types of Petri Nets
- Elements of CPN
- Why CPN
- CPN Tools

2.1 Introduction

- Petri nets were developed in the early 1960s by C.A. Petri in his Ph.D. dissertation
  o C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut
  for instrumentelle Mathematik, Bonn, 1962 [8].
- Petri Nets are graphical and mathematical modeling tool for describing and studying information processing systems that are characterized as being concurrent, synchronous, distributed and non-deterministic [7].

Predicate/Transition Nets(Pr/T-nets):-

- A Petri net consists of *places*, *transitions*, and *directed arcs*. Arcs are either from a place to a transition or from a transition to a place, never between places or between transitions. The places from which an arc runs to a transition are called the *input places* of the transition; the places to which arcs run from a transition are called the *output places* of the transition [7].
- Places may contain any non-negative number of *tokens*. A distribution of tokens over the places of a net is called a *marking*.
- A Petri Net is a 5 tuple

  PN = (P,T,F,W,M0),

  where

  P = {p1, p2, … pm} is a finite set of places

  T = {t1, t2, …, tn} is a finite set of transitions

  F Í (P X T) È (T X P) is a set of arcs

  W: F® {1,2,3,…} is a weighting function

  M0 :P® {0,1,2,…} is the initial marking // defines

  Number of tokens per place [7].

Example:-

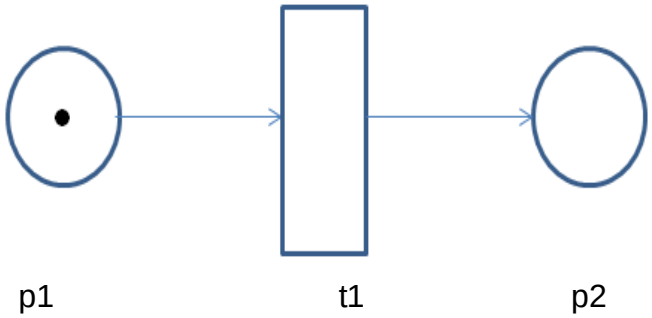

Fig-1.1: showing a basic Petri net [7]

Description of the above figure:

Places- p1, p2 Transitions-t1

M(p1)=1 , i.e.- p1 has one token

M(p2)=0 , i.e.- p2 has no token

Firing of a transition:-

- A transition of a Petri net may *fire* whenever there is a token at the end of all input arcs; when it fires, it consumes these tokens, and places tokens at the end of all output arcs [7].
- A firing is atomic, i.e., a single non-interruptible step. Execution of Petri nets is nondeterministic: when multiple transitions are enabled at the same time, any one of them may fire [7]. If a transition is enabled, it may fire, but it doesn't have to.
- Since firing is nondeterministic, and multiple tokens may be present anywhere in the net (even in the same place), Petri nets are well suited for modeling the concurrent behavior of distributed systems [7].

- In order to simulate the dynamic behavior of a system, a state or marking in a Petri nets is changed according to the following *transition(firing) rule*.

Transition rule:-

I. A transition t is said to be *enabled* if each input place p of t is marked with at least w(p,t) tokens , where w(p,t) is the weight of the arc from p to t [2].

II. An enabled transition may or may not fire(depending on whether or not the event actually takes place) [2].

III. A firing of an enabled transition t removes w(p,t) tokens from each input place p of t and adds w(t,p) tokens to each output place p of t, where w(t,p) is the weight of the arc from t to p [2].

Basing upon these firing conditions Petri nets can be of 2 types:

1.Original Petri nets

2. Weighted Petri nets

2.2 Original Petri Nets:

- Only 1 token can be removed/added from a place when a transition fires(i.e., the weight of the arc is always 1) as shown in the figure 2.1(a) and 2.1(b) [8].

Example:

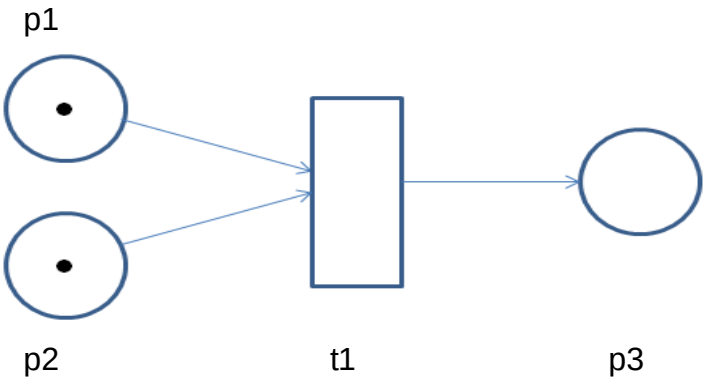

Fig-2.1(a): Petri net before t1 fires

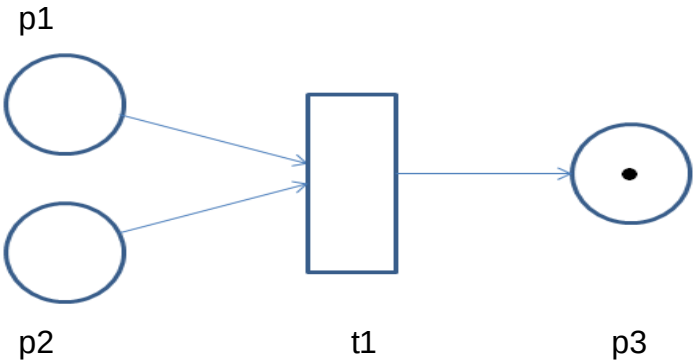

Fig-2.1(b): Petri net after t1 fires

2.3 Weighted Petri Nets:

- Generalized the original Petri net to allow *multiple tokens* to be added/removed when a transition fires as shown in the fig. 2.2.
- The edges are labeled with the weight (i.e., number of tokens).
- If there is no label, then the default value is 1 [8].

Example:

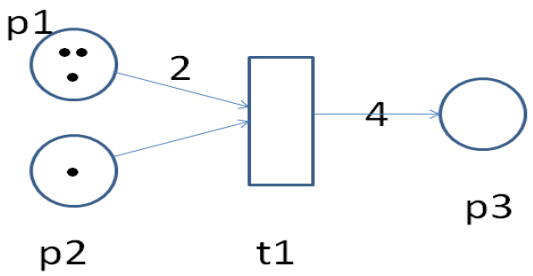

Fig-2.2(a): Petri net before t1 fires

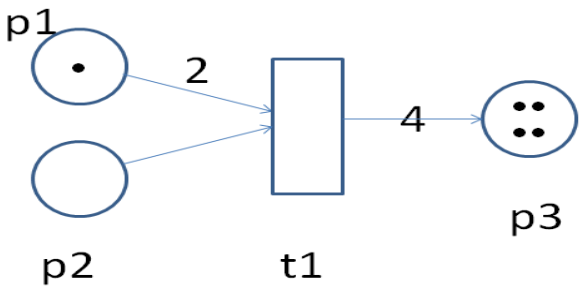

Fig-2.2(b): Petri net after t1 fires

2.4 Other Types Of Petri Nets:

Petri nets have been extended over the years in many directions including time, data, and hierarchy.

Time Extended Petri nets

- First developed in the mid 1970s [3].
- For real systems it is often important to describe the *temporal behavior* of the system, i.e., we need to model durations and delays.
- There are 3 basic ways to introduce time into the Petri net. Time can be associated with:
  - tokens
  - places
  - transitions

The first introduction of time in Petri nets is in the *Timed Petri net model* [3]

- C. Ramchandani, "Analysis of Asynchronous concurrent systems by timed Petri nets", MIT MAC-TR-120,1974 [3].
- In this model, a time duration is associated with each transition.
- The firing rules in this model are that the transition must fire as soon as it is enabled and firing a transition takes a fixed, finite amount of time.
  - The notion of instantaneous firing of transitions is not preserved in the Timed Petri net model.
  - When a transition becomes enabled, the tokens are immediately removed from its input places.
  - After the time delay, tokens are deposited in the output places.
  - The result is that the state of the system is not always clearly represented during the process.

Colored Petri Nets

- Developed in the Late 1970s
  - K. Jensen, “Colored Petri nets and the invariant method”, Theoretical Computer Science, volume 14,1981,pp. 317-336 [2].
- CP-net has 3 parts:
  I. Net structure(places, transitions, arcs)
  II. Declarations
  III. Net inscriptions(text strings attached to the elements of net structure).
- Places, transitions and arcs together constitute the *net structure.*
- Textual strings next to the places, transitions and arcs are written in CPN ML [4].
- Each place can be marked with one or more tokens.
- Tokens often represent objects (e.g. resources, goods, humans) in the modeled system [5].
- To represent attributes of these objects, the Petri net model is extended with *colored or typed tokens* [2].
  - each token has a data value attached to it often referred to as *‘token color’*.
- It is the number of tokens and the token colors on the individual places which together represent the state of the system. It is called marking of the CPN model.
- Transitions use the values of the consumed tokens to determine the values of the produced tokens [8].
  - a transition describes the relation between the values of the `input tokens' and the values of the ‘output tokens’.
- It is also possible to specify `preconditions' which take the colors of tokens to be consumed into account.

Hierarchical Petri Nets

- Developed in the Late 1970s by Valette.
- R. Valette, “Analysis of Petri Nets by Stepwise Refinement,” in *Journal of Computer and System Sciences,*vol. 18, pp. 35-46, Feb.1979 [2].
- Specifications for real systems have a tendency to become large and complex.
- An abstraction mechanism, hierarchical structuring, is used to make constructing, reviewing and modifying the model easier.
- The hierarchy construct is called a *subnet.*
- A subnet is an aggregate of a number of places, transitions, and subsystems.
- Such a construct can be used to structure large processes [9].
- At one level we want to give a simple description of the process (without having to consider all the details). At another level we want to specify a more detailed behavior.
- Each subnet is represented with a rectangular box that encapsulates part of the Petri net model as shown in the figure 2.3.

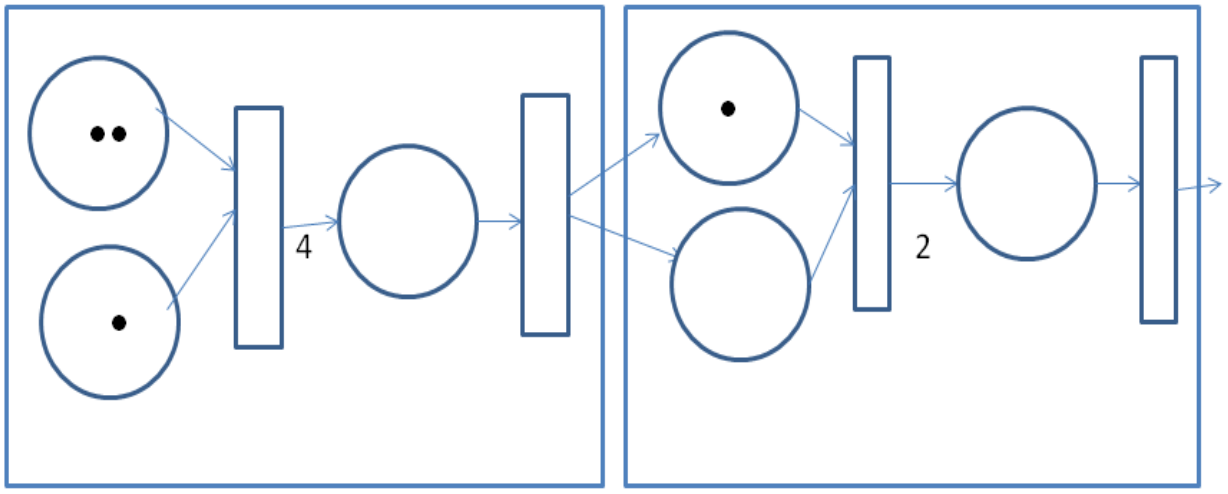

Fig. 2.3- A Hierarchical Petri Net

2.5 Elements of CPN:

- CPN has 3 parts:
  - net structure
  - declarations
  - net inscriptions
- Places, transitions and arcs together constitute the net structure.
- The ellipses and circles are called *places .*They describe the state of the system.
- The rectangles are called *transitions*. They describe the actions [7].
- The arrows are called *arcs*. The arc expressions describe how the state of the CP-net changes when the transitions occur [5].
- Each place contains a set of markers called *tokens*. Each of these tokens carries a data value, which belongs to a given type, known as *token color*.

  In section 2.6, we have discussed why colored Petri net is more suitable than any other tool for implementing the algorithm.

2.6 Why Colored Petri Nets:

- CP-nets have a graphical representation.
- CP-nets are very general and can be used to describe a large variety of different systems.
- They have an explicit description of both states and actions.
- CP-nets offer hierarchical descriptions.
- They also offer interactive simulations where the results are presented directly on the CPN diagram [5].
- CP-nets have computer tools supporting their drawing, simulation and formal analysis.

Real world systems contain many parts which are similar but not identical which are represented by disjoint subnets of PT-nets [2]. So PT-net becomes large and difficult to see the similarities between the individual subnets representing similar parts. Colored Petri net is a discrete-event modeling language combining Petri nets with the

functional programming language standard ML. CPN is a language for the modeling and validation of systems in which concurrency and synchronization play a major role.

A CPN model of a system is an executable model representing the states of the system and the events(transitions) that can cause the system to change state. By making simulations of the CPN model [2] it is possible to investigate different scenarios and explore the behaviors of the system. In the next section 2.7 we have discussed about the CPN tools which we have used for implementation of the algorithm.

2.7 CPN Tools:

CPN tools is a tool for editing, simulating and analyzing colored Petri nets.

- The GUI is based on advanced interaction techniques, such as toolglasses, marking menus and bimanual interaction.
- Feedback facilities provide contextual error messages and indicate dependency relationships between net elements.
- The tool features incremental syntax checking and code generation which take place while a net is being constructed. A fast simulator efficiently handles both untimed and timed nets [4].
- Full and partial state spaces can be generated and analyzed and a standard state space report contains information such as boundness properties and liveness properties [5].
- The functionality of the simulation engine and state space facilities are similar to the corresponding components in Design/CPN, which is a widespread tool for colored Petri nets [4].

Chapter-3

MOBILE AD-HOC NETWORK

- *Introduction*
- *Mobility Problem*
- *Related Work*
- *Proposed algorithm*
- *Results*

3.1 Introduction

A mobile ad hoc network(MANET) is a kind of wireless network and is a self configuring network of mobile routers(and associated hosts) connected by wireless links- the union of which forms an arbitrary topology [1]. The routers are free to move randomly and organize themselves arbitrarily, thus the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion or may be connected to the larger internet. Mobile adhoc networking is a technology for the co-operative engagement of a group of mobile nodes without requiring established infrastructure and centralized administration.

MANET nodes are equipped with wireless transmitters and receivers using antennas which may be omnidirectional (broadcast), highly- directional (point-to-point), possibly steerable, or some combination thereof [6]. At a given point in time, depending on the nodes' positions and their transmitter and receiver coverage patterns, transmission power levels and co-channel interference levels, a wireless connectivity in the form of a random, multihop graph or "ad hoc" network exists between the nodes [1]. This ad hoc topology may change with time as the nodes move or adjust their transmission and reception parameters.

MANETs have several salient characteristics:

1) Dynamic topologies: Nodes are free to move arbitrarily; thus, the network topology--which is typically multihop--may change randomly and rapidly at unpredictable times, and may consist of both bidirectional and unidirectional links.

2) Bandwidth-constrained, variable capacity links: Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications--after accounting for the effects of multiple access, fading, noise, and interference conditions, etc.--is often much less than a radio's maximum transmission rate [9].

   One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will

likely approach or exceed network capacity frequently. As the mobile network is often simply an extension of the fixed network infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise [9].

3) Energy-constrained operation: Some or all of the nodes in a MANET may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation [9].
4) Limited physical security: Mobile wireless networks are generally more prone to physical security threats than are fixed-cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in MANETs provides additional robustness against the single points of failure of more centralized approaches [6].

In a MANET the nodes move at their will thus changing the topology of the MANET dynamically. This mobility problem is discussed in the next section 3.2.

3.2 Mobility Problem:

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 [1]. In a MANET if node A lies within the transmission range of another node B, we say there is a link(edge) between them in the graph describing the MANET, where node A is called a neighbor of node B and vice versa.

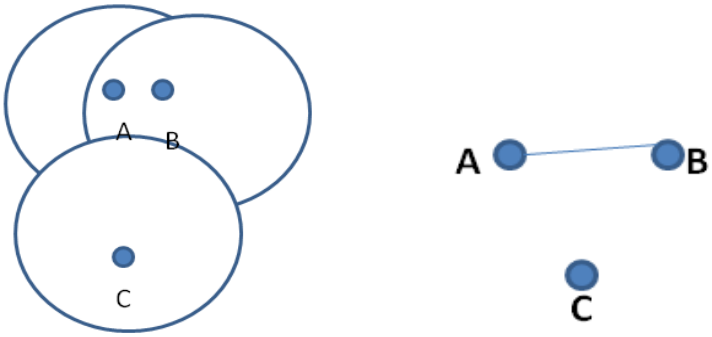

Fig.3.1(a)- Nodes A and B are neighbors [1]

For example, in Fig.3.1(a), nodes A and B are neighbors since they are within the transmission ranges(shown by circles) of each other but node C is not a neighbor of node A since node C is not within the transmission range of node A.

Every node is allowed to move at will in a MANET and thus a link between nodes may disappear and reappear in an unpredictable manner. For example in Fig.3.1(a) when node A moves out of the transmission range of node B, the link between nodes A and B breaks as shown in the Fig.3.1(b). When node B moves back within the transmission range of node A, the link between them reappears.

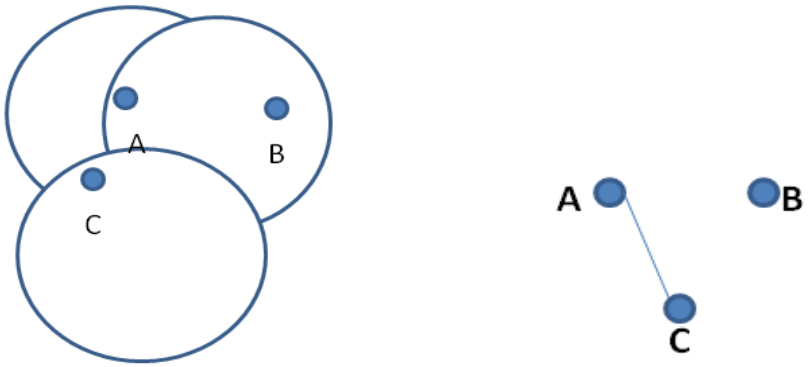

Fig.3.1(b)- Node A moves out of the transmission range of node B moves into C

In an extreme case the graph of a MANET remains the same even if all nodes have moved; e.g., the MANET graphs in Fig.3.1(a) and 3.1(c) are the same, although all three nodes have moved from their original positions. Therefore, it is reasonable to disregard the exact locations or movements of nods when we build a CPN model for a MANET.

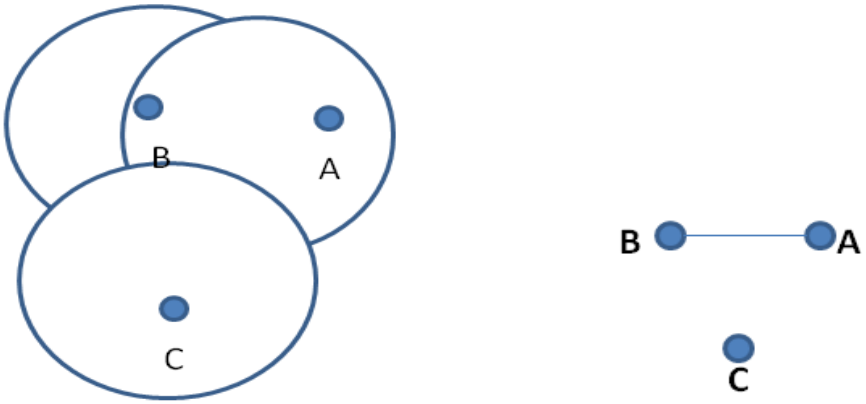

Fig.3.1(c): The topology remains the same as that in Fig.5(a) although all three nodes have moved [1].

Nevertheless, to verify the performance of a routing protocol, a CPN model still should simulate the mobility of a MANET, while it is difficult to build a dynamic structure in CPN modeling. In addition, to find a route from a source node to a destination, the source node in the CPN model should have its neighbors' identifications to send messages. It is difficult to capture structure changes in a MANET because the structure of a MANET changes in an unpredictable way [1]. Nodes in a MANET move at will and their movements change neighboring relationships unpredictably. Therefore, it is not easy to build a CPN model of a MANET as nodes can move in and out of their transmission ranges and thus MANET's topology(graph) dynamically changes. To address this problem we have discussed topology approximation, that has already implemented, algorithm in section 3.3 and the proposed algorithm in section 3.4.

3.3 Related Work:

To address the mobility problem, a topology approximation(TA) [1] algorithm has been already proposed in building CPN models for a routing protocol. The TA mechanisms can simulate the mobility of a MANET without any loss of useful information that represents the semantics and performance of routing protocols used in MANETs. Even if neighbors' identifications are recorded at every node for a reactive protocol, this information on neighbors is time-varying and may be inaccurate after a certain time because nodes can move. The selection of a route across a large or set of networks is typically performed using only partial or approximate information. In the TA mechanism proposed, uses a receiving function for selecting sender's neighbors which receive messages sent by a sender node, in order to mimic the dynamic structure of a MANET.

Topology Approximation Algorithm:

The TA mechanism [1] uses a receiving function to describe the dynamic structure of a MANET. It 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 no of nodes, the average no of neighbors of a node will depend on the area it covers: If all nodes are dispersed in a small area, the average number of neighbors will be larger than that when nodes are dispersed in a wider area. Also if every node has the same responsibility and can move within a fixed area at will, the average number of neighbors for a node is nearly constant ,if there are a fixed number of nodes within that area. Thus average number of neighbors can well represent the approximate topology in formation of a MANET. Based on this observation the TA mechanism uses a receiving function to decide which node receives which message [1].

Assume every node in a MANET has the same number of neighbors which is equal to average degree of MANET graph. Since every node has the same capacity and responsibility, every node in a MANET has the same chance of receiving or forwarding broadcast message [1]. Broadcast messages are sent or forwarded by a node through radio waves. In a real situation, neighboring nodes will directly receive a broadcast message sent by this node. In CPN modeling, it creates a place called *Store*

to hold all the message in transmission [1]. After the place Store receives messages ,a receiving function directs messages to the corresponding neighboring node. 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 node's neighbor will receive broadcast message sent by the node. Thus a node should maximally send (n-1) copies of a broadcast message to the place Store . For convenience in CPN modeling ,we always choose (n-1) as the number of copies of the broadcast message sent or forwarded by a node. But only x out of these (n-1) copies of the broadcast message will 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 message will be received by other nodes is 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 [1]. Thus it results:

PB =d/[(n-1)(n-1)]

Nodes in a MANET also receive unicast message such as route reply(RREP) message, which are different from broadcast message such as route request (RREQ) message. Thus the receiving function needs to check if a message is a unicast or broadcast message.

```
If(the message is of type RREQ and is not sent by the node){
        Generate a random integer K, between a and 100;
        If(100PB>K)
                The node will receive the message;
        else
                The node will not receive the message;}
else if(the message is of type RREP and is for the node)
        the node will receive the message;
else    the node will not receive the message;
```

3.4 Proposed Algorithm:

In this paper we have proposed an algorithm for selecting sender's neighbors which receive messages sent by a sender node. Each node has several parameters like: node id, transmission range, velocity, neighbors set etc. In our proposed algorithm we have made two assumptions:

1. The transmission range of each node is constant;
2. Each node moves with the same velocity.

The basic idea is that we can construct a circle by taking the transmission range as radius. For each node, which wants to send a message by knowing their neighbors, we can construct a circle. The nodes that are inside the circle are the neighbor nodes and these nodes are entered in the neighbor set of the sender node. So each time before updating the neighbor list we have to see the nodes that are coming within the area of the circle.

Whenever a node wants to know its neighbor, it broadcasts a "HELLO" packet . From the transmission range and velocity we can calculate the time required by the hello packet to reach at their neighbors. Upon receiving the hello packet the neighbors update their neighbor list and sends an acknowledgement to the sender. Since the transmission range and velocity are constant quantities, we can calculate the time required by the hello packet to reach at the neighbor. Also the acknowledgement from the neighbors must arrive at the sender side by 2t times, where t is equal to time taken for the hello packet to reach at their neighbor. All the nodes maintain a neighbor set. Each node updates its corresponding neighbor set by receiving the acknowledgement from the neighbors. Thus the neighbor set is updated continuously as shown in the figure 3.2.

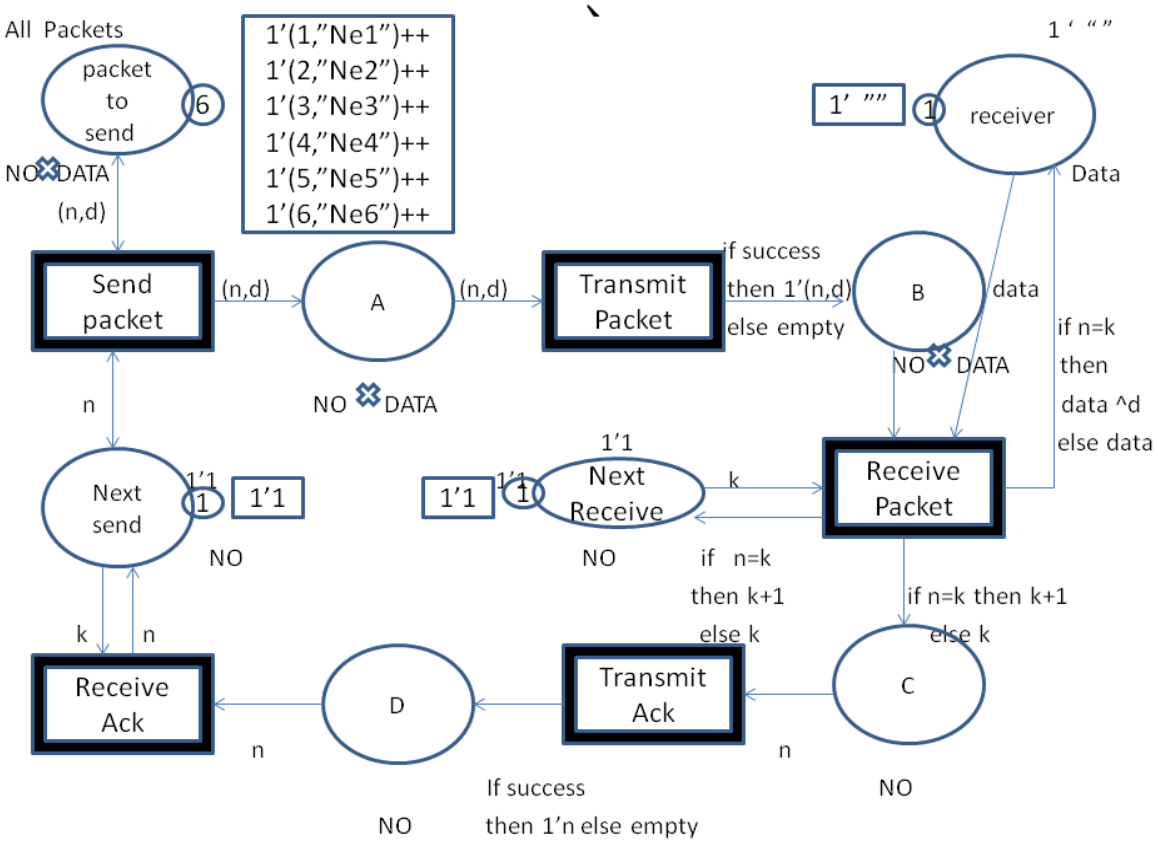

Fig.3.2: Detection of neighbors in the MANET

The state of the sender is modeled by the two places PacketsToSend and NextSend. The state of the receiver is modeled by the two places DataReceived and NextRec, and the state of the network is modeled by the places A, B, C, and D. Next to each place, there is an inscription which determines the set of token colors (data values) that the tokens on the place are allowed to have. The set of possible token colors is specified by means of a type (as known from programming languages), and it is called the color set of the place. By convention the color set is written below the place. The places NextSend, NextRec, C, and D have the color set NO. In CPN Tools, color sets are defined using the CPN ML keyword colset, and the color set NO is defined to be equal to the integer type int:

colset NO = int;

This means that tokens residing on the four places NextSend, NextRec, C, and D will have an integer as their token cooler. The color set NO is used to model the sequence numbers in the protocol. The place DataReceived has the color set DATA defined to be the set of all text strings string. The color set DATA is used to model the payload of data packets. The remaining three places have the color set NOxDATA which is defined to be the product of the types NO and DATA. This type contains all two-tuples (pairs) where the first element is an integer and the second element is a text string. Tuples are written using parentheses (and) around a comma separated list. The color set NOxDATA is used to model the data packets which contain a sequence number and some data. The color sets are defined as:

colset DATA = string;

colset NOxDATA = product NO * DATA;

Next to each place, we find another inscription which determines the initial marking of the place. The initial marking inscription of a place is by convention written above the place. For example, the inscription at the upper right side of the place NextSend specifies that the initial marking of this place consists of one token with the colour (value) 1. This indicates that we want data packet number 1 to be the first data packet to be sent. Analogously, the place NextRec has an initial marking consisting of a single token with the color 1. This indicates that the receiver is initially expecting the data packet with sequence number 1. The place DataReceived has an initial marking which

consists of one token with color "" (which is the empty text string). This indicates that the receiver has initially received no data and its neighbor list is empty The ++ and ‘ are operators that allow for the construction of a multi-set consisting of token colors. A multi-set is similar to a set, except that values can appear more than once. The infix operator ‘ takes a nonnegative integer as left argument specifying the number of appearances of the element provided as the right argument. The ++ takes two multi-sets as arguments and returns their union (sum). The initial marking of PacketsToSend consists of six tokens representing the data packets which we want to transmit. The data contained in the packet includes the name of the neighbors. The absence of an inscription specifying the initial marking means that the place initially contains no tokens. This is the case for the places A, B, C, and D.

The current marking of each place is indicated next to the place. The number of tokens on the place in the current marking is shown in the small circle, while the detailed token colors are indicated in the box positioned next to the small circle. Initially, the current marking is equal to the initial marking, denoted M0. As explained earlier, the initial marking has six tokens on PacketsToSend and one token on each of the places NextSend, NextRec, and DataReceived.

The five transitions (drawn as rectangles) represent the events that can take place in the system. As with places, we write the names of the transitions inside the rectangles. The transition names also have no formal meaning but they are very important for the readability of the model. When a transition occurs, it removes tokens from its input places (those places that have an arc leading to the transition) and it adds tokens to its output places (those places that have an arc coming from the transition). The colors of the tokens that are removed from input places and added to output places when a transition occurs are determined by means of the arc expressions which are the textual inscriptions positioned next to the individual arcs. A transition and a place may also be connected by double-headed arcs. A double-headed arc is shorthand for two directed arcs in opposite directions between a place and a transition which both have the same arc expression. This implies that the place is both an input place and an output place for the transition. The transition SendPacket and the places PacketsToSend and NextSend are connected by double-headed arcs. The arc

expressions are The arc expressions are written in the CPN ML programming language and are built from typed variables, constants, operators, and functions. When all variables in an expression are bound to values (of the correct type) the expression can be evaluated. An arc expression evaluates to a multi-set of token colors. As an example, consider the two arc expressions: n and (n,d) on the three arcs connected to the transition SendPacket. They contain the variables n and d declared as:

var n : NO;

var d : DATA;

This means that n must be bound to a value of type NO (i.e., an integer), while d must be bound to a value of type DATA (i.e., a text string).

Arc expressions evaluate to a multi-set of token colors, and this means that there may be zero, exactly one token, or more than one token removed from an input place or added to an output place. If an arc expression evaluates to exactly one token, then the 1‘ can be omitted from the expression by convention. For example, arc expressions n and (n,d) are shorthand for 1‘n and 1‘(n,d).

The arc expressions on the input arcs of a transition together with the tokens on the input places determine whether the transition is enabled. If the transition Send Packet is enabled then it fires resulting the hello packet to be transmitted in the network. Then the nodes receiving the hello packet updates its neighbor list and sends an acknowledgement to the sender. The sender then updates the neighbor set and again sends back an acknowledgement to the receiver that it has added it to its neighbor list.

3.5 Results:

The results obtained by simulating the above algorithm are shown below in the figure 3.3(a), 3.3(b) and 3.3(c).

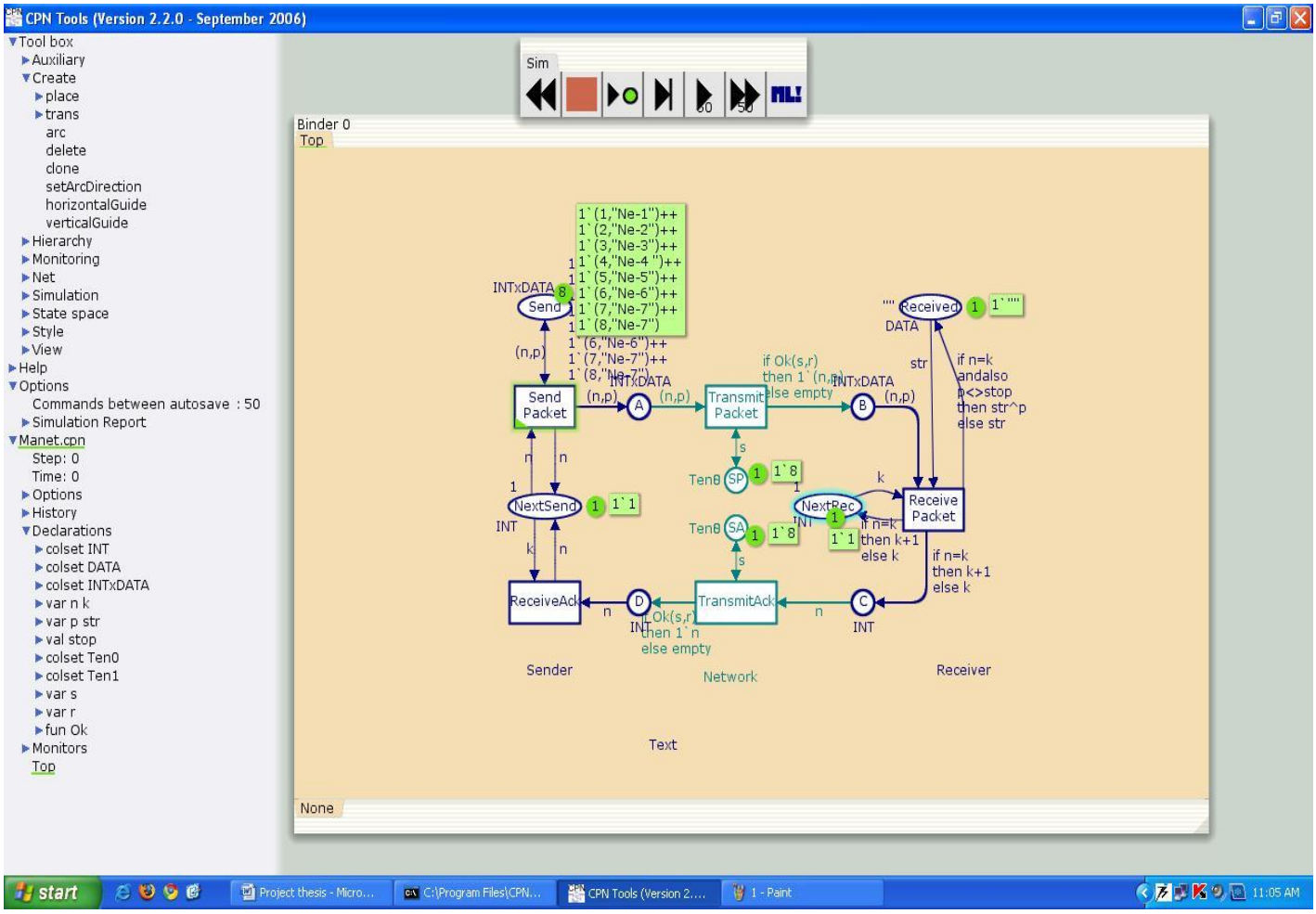

Fig. 3.3(a): Snapshot of the whole transmission at the starting state

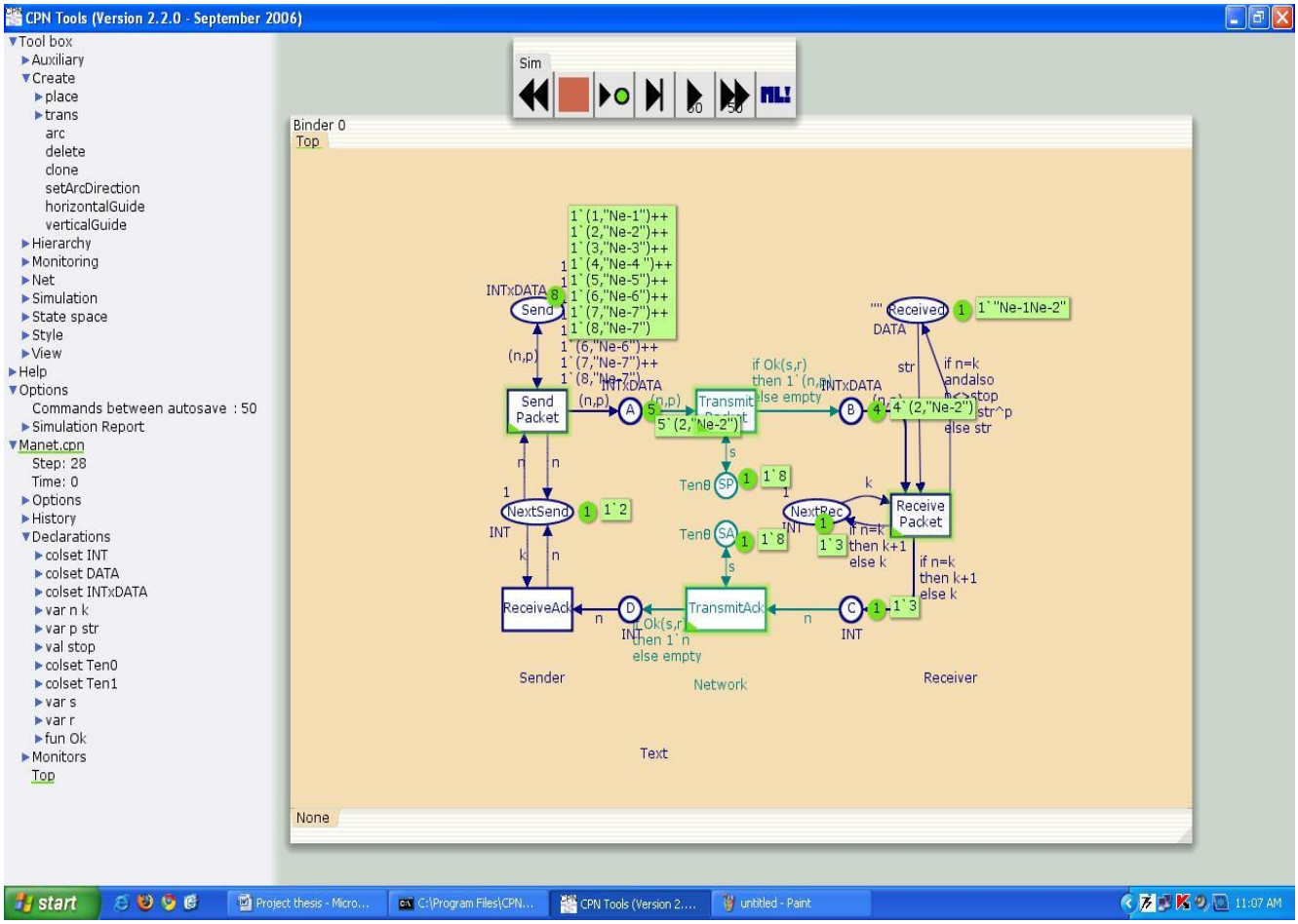

Fig.3.3(b): Snapshot of the updation of the neighbor set

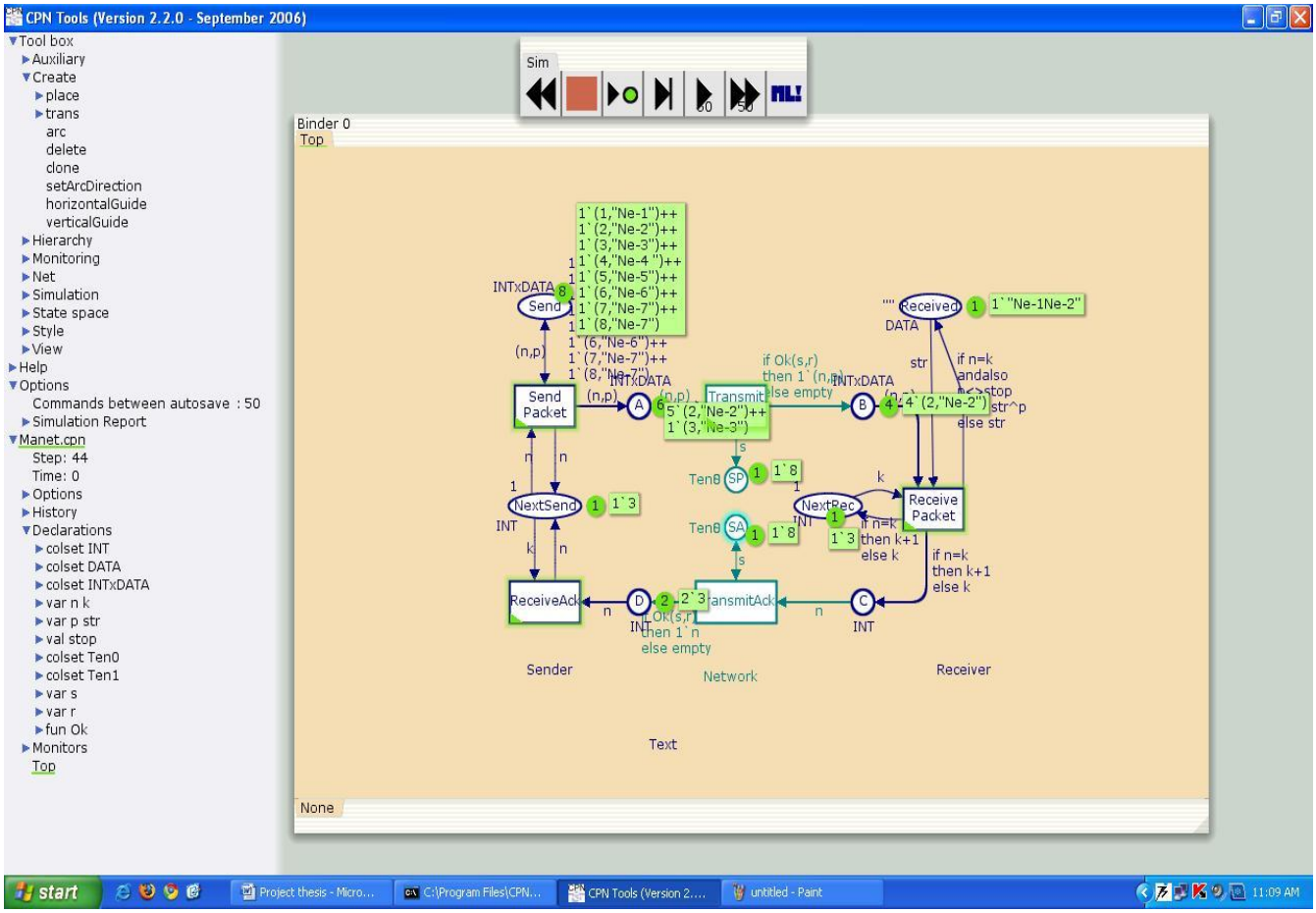

Fig.3.3(c): Snapshots while multiple receivers send their acknowledgement.

Chapter-4

COMPARISON BETWEEN THE TOPOLOGY APPROXIMATION ALGORITHM AND THE PROPOSED ALGORITHM

Comparison:

The topology approximation mechanism can simulate the mobility of a MANET without any loss of useful information that represents the semantics and performance of the routing protocols used in MANETs. It does not give the exact neighbors of a node rather it uses a receiving function based on which each node decides whether to receive or reject the broadcast message. If it is a route reply message and is intended for the sender then the sender accepts the message observing its address on the reply message. If it is a request message then basing upon the probability bar function it decides whether to accept or reject the message [1]. The TA mechanism simulate the mobility(dynamically changing graphs) of a MANET without knowing its actual topology .

In contrast to it, the proposed algorithm sends a message in two steps. In the first step it finds out the neighbors and then it sends the message to its neighbors. So overhead becomes more than that of topology approximation algorithm but exact neighbors can be identified. Again if a message or an acknowledgement is lost on the way due to congestion or network failure then the node cannot identify that node as a neighbor whose acknowledgement is lost although it is actually a neighbor of it. So under very high load conditions due to network congestion, topology approximation algorithm outperforms the proposed algorithm.

Chapter-5

CONCLUSION

Conclusion:

With the explosive growth of the Internet and mobile communication networks, challenging requirements have been added into MANETs and designing routing protocols has become more and more complex. Unexpected combinations of events can drive protocols into undesirable states, affect the protocol performance and even lead protocols to errors. For a successful application of MANETs, it is vitally important to ensure that a routing protocol is unambiguous, complete and functionally correct. One approach to ensure correctness of an existing routing protocol is to create a formal model for the routing protocol and analyze the model if indeed the protocol provides the defined service correctly. CPNs are a suitable modeling language for this purpose as they can conveniently express non-determinism, concurrency and different levels of abstraction that are inherent in routing protocols. 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 MANET's topology changes dynamically. So it is important to detect the neighbors of a node to which the sender can send a message.

This paper contains two algorithms for detecting the neighbors of a node: one is the topology approximation algorithm that is implemented already and another one is our proposed algorithm. The TA algorithm does not give us idea about the exact neighbors of a node rather it uses a receiving function for each node. Each node calculates the receiving function basing upon which it decides whether to accept or reject the message. Our proposed algorithm detects the exact neighbors of a node but it encounters a greater overhead than that of the TA algorithm. Again in high load conditions, when there is congestion in the network, some of the hello packets or acknowledgements may be lost so that a neighbor of a node is undetected due to missing of the acknowledgements. So under high load conditions the TA outperforms the proposed algorithm.

Chapter-6

REFERNCES

References:

[1]. Chaoyue Xiong, Tadao Murata, Jeffery Tsai, "*Modeling and Simulation of Routing Protocol for Mobile Ad Hoc Networks Using Colored Petri Nets*", Department of Computer Science, University of Illinois at Chicago,USA,2005, pages 1-4.

[2]. K. Jensen and L. M. Kristensen. "*Coloured Petri Nets, Modelling and Validation of Concurrent Systems*" Springer-Verlag. Textbook, in preparation., pages 8-32, 174-192.

[3]. S. Christensen, L. M. Kristensen, and T. Mailund. "*Condensed State Spaces for Timed Petri Nets. In Proc. Of International Conference on Application and Theory of Petri Nets"*, volume 2075 of Lecture Notes in Computer Science, Springer-Verlag, 2001, pages 101-120.

[4]. CPN Tools. www.daimi.au.dk/CPNTools/.

[5]. K. Jensen. "*Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use"*. Volume 1&2: Basic Concepts. Springer-Verlag, 1992, pages 12-23, 87-103.

[6]. J. Billington, M. Diaz, and G. Rozenberg, editors. *"Application of Petri Nets to Communication Networks"*, volume 1605. Springer-Verlag, 1999.

[7]. http://en.wikipedia.org/wiki/Petri_net#Petri_nets_basics.

[8]. Lars M. Kristensen, S_ren Christensen, Kurt Jensen. *"The practitioner's guide to coloured Petri nets"***.** CPN Group, Department of Computer Science, University of Aarhus, Denmark, 1998.

[9]. Lars Michael Kristensen_, Jens Bæk Jørgensen, and Kurt Jensen*."Application of Coloured Petri Netsin System Development".* Department of Computer Science, University of Aarhus IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark.