Simulation of Dynamic Mobile Agent Model to Prevent Denial of Service Attack using CPNS

Mayank Aggarwal
Department of Computer Science & Engineering, Gurukula Kangri University, Haridwar, 249404, India

Nupur
Department of Computer Science, Kanya Gurukul Mahavidyalaya, Sewak Ashram Road, Dehradun, 248001, India

Pallavi Murgai
Department of Computer Science, Kanya Gurukul Mahavidyalaya, Sewak Ashram Road, Dehradun, 248001, India

ABSTRACT
Mobile Agents are soft wares migrating from one node to another to fulfill the task of its owner. In Static mobile agents, agent travels on the predefined path whereas in Dynamic mobile agents, agent route is decided by the host or the agent itself, which makes security much more difficult in it. Mobile agents are not properly utilized because of security concerns. Security becomes more challenging in Dynamic mobile agents as compared to Static mobile agent. One such challenge is 'Denial of Service', in it the malicious host may deny resources required by the agent and kill the agent, thus the result computed so far is lost and this may happen every time the agent visits any malicious host. Colored Petri Nets (CPNs) is a language for the modeling and validation of systems in which concurrency, synchronization and communication play a major role. In our previous paper we have simulated and obtained the results for static mobile agent but in real world agents are dynamic. This paper simulates dynamic mobile agent model that enables the owner of the agent to detect the malicious host. The simulation has been done using CPNs, the result clearly proves that owner can detect the malicious hosts and thus prevent Denial of service attack to occur in future

Keywords
Dynamic mobile agent; denial of service attack; colored petrinets; independent and dependent mobile agent..

1. INTRODUCTION
The advantage for using mobile agent technology is that interaction cost for the agent owner is remarkably reduced since after leaving its owner the agent migrates from one host to the next autonomously. Though a lot of research is going for security of mobile agents and host security [14], it is still a major concern. One such concern is 'Denial of Service' attack by malicious host, in such an attack; the malicious host can prevent an agent from migrating to another host or may even delete the agent. As a consequence, all the results collected so far are lost. This may repeat every time the agent passes through this malicious host while the agent owner has no knowledge to detect the malicious host[12]. In general, the term Denial Of Service is used for attacks in which the focus is on exhausting resources with the effect that other entities cannot be served anymore[12]. Mobile agents are of two types Static and Dynamic. Static mobile agent follows the route decided by its owner. Dynamic mobile agent the route is decided by the agent or the host at each transition. The security concern increases in case of Dynamic mobile agent as the chances of host to be visited next for being malicious is more as compared to the hosts in static mobile agent because in it the host is not predicted by the owner.This paper is an extension of the work done in 'Simulation of Static mobile agent to prevent denial of service of service attack using CPNs' **(Aggarwal Mayank, Nupur,Pallavi 2011) in press**, a similar model was discussed in [11,12], additional and useful changes in the model have been done and model is finally simulated to show that owner can detect the malicious host. The paper discusses how the owner can detect the malicious host in dynamic mobile agent and thus prevent from Denial Of Service attack in future .The paper is divided in XI Sections. In section II we formally define the agent, data structure and the problem. Section III discusses the solution whose algorithm is taken in Section IV with algorithms for sender, receiver ,router and guard. Section V has the model showing the owner, hosts ,router and the guard and how do they communicate with each other. Section VI discusses the detection mechanism i.e. how the owner detects the malicious host. Implementation of the model in CPNs is discussed in section VII. The simulation using CPN is discussed in section VIII showing different markings. Many researchers have simulated mobile agents in CPN tools before also [13]. The limitations of simulation is overcome in Section IX by State Space Analysis of the model.The related work on agent security is discussed in Section X. Last Section i.e. Section XI has Conclusion and scope for future. The paper shows screen shots of the simulation done in CPN tool, for clarity of pictures only the required portion is captured thus the label not required appear truncated.

2. DEFINING AGENTS AND PROBELM
Agent is a software program which works on behalf of the user. There are many possible definitions of mobile agent, **Lang C (1998) at General Magic, Inc: said** agent is a software object that is situated within an execution environment and must **possess the following mandatory properties":**

a) Autonomous : On behalf of the user
b) Reactive : Responsive to change in environment.
c) Goal Driven : Proactive acting in advance to deal with an expected situation.
d) Temporally Continuous: Continuously operating

In this paper agent is defined as:

$Agent^j$ = (id, sc, r,) i.e agent at j^{th} host.

Where:

j : Stands for the host i.e. j^{th} host

id : It is the unique identifier of the agent.

sc : It denotes the source code of the agent.

r : Predefined route(set of trusted hosts)

j^{th} host is denoted by c_j and it is assumed that agent travels from c_j to c_x where c_x is decided dynamically.

There can be two types of operations in agent systems: *Dependent* and *Independent* [12].

In a *dependent* computation (the output of one host is input to next host) Here, at c_j the results contained in *o/p (j-1)* calculated by c_{j-1} are needed as input for c_j. Thus the order of hosts in the journey is important.

We say that an agent journey is *completely dependent* if for $1 < i \leq n$ each c_i requires the results of c_{i-1}. In this case, the hosts have to be visited exactly in the same order as they are prescribed in the route *r*.

This means that the agent cannot fulfill its task when at least one of the required hosts specified in *r* is either not available or denies its services to the agent.

Second type is *independent* (in which the order of journey is not important) Here, the agent computation does not need the results of another host. We say that an agent journey is *completely independent* when there are no dependent computations in it. As a consequence, the hosts contained in the prescribed route r can be visited in any arbitrary order.

The agent is free to move without the control of its owner, therefore the possibilities of its loss increases a lot. As the agent moves from c_j to c_x, the possibility of c_x being malicious is significant and thus 'Denial of Service' attack might occur, which would result in loss of the agent and the partially computed result. We can also say, In 'Denial Of Service' the agent is not transmitted further from the malicious host and the owner keeps on waiting for the results and the agent.

In the next section we provide a solution for the discussed problem.

3. SOLUTION

A model is proposed which can solve the problem discussed in section 2. In the proposed model owner could detect the malicious host and thus prevent from 'Denial of Service' attack to occur in future journey.

The model has two main servers called 'Guard' and 'Router'. Router keeps track of the dynamically decided route and Guard keeps all the acknowledgements.

For each value in the router there should be an acknowledgement with the guard if agent is successfully transmitted.

Whenever an agent moves from host c_j to c_x (decided dynamically) the host c_j sends an acknowledgment **"j"** to the guard and sequence of next host **"x"** to the router.

Whenever the owner feels that agent has not returned it checks with the router for the route and for their corresponding acknowledgments in the guard , if any acknowledgment is missing it means that agent has not been transmitted correctly and the owner can detect the malicious host. For example if route is [1, 3, 7] and acknowledgement form host 3 is missing, then it suggests that host 3 might be malicious.

It may also be the case that that c_j is ready to send but c_x is found to be offline always. This variation is dealt separately by using the concept that c_x should be blacklisted from the trusted node lists and c_j should avoid sending agent to c_x . Although this raises another issue that malicious c_j can malign the image of c_x and skip it intentionally though it is available.

The algorithm proposed gives the solution of the simplified problem not the above variation.

Many protocols like in [8] can be reused, [8] discusses how to implement cryptographic protocols and reuse it. The proposed architecture in [8] can be used for implementation of secure distributed applications.

4. ALGORITHM

The model broadly has three main players 'Sender', 'Router' and 'Guard'. Role of 'Receiver' is same as 'Sender' as it becomes the Sender for next host.

It is assumed that c_j sends agent to c_x. (Figure 1)

4.1 Sender Algorithm

1.Saves a copy of agent

2.Send the agent to c_x. Where c_x is dynamically decided.

3. If successfully sends the agent to c_x, sends acknowledgement bearing $_j$ to guard.

4. Send the signature x of c_x the next host to router.

4. End

The sender saves a copy of agent so that if c_x is malicious and it changes the code , a copy of it and the result up to host j can be had from c_j.

4.2 Receiver Algorithm

The receiver works similarly as sender. The receiver on receiving the agent becomes sender for next host in the path. It follows the same algorithm as the sender.

4.3 Guard Algorithm

In our model, guard plays an important role. We can consider it as a server keeping all records; similar to the server it keeps copies of all acknowledgments for future reference(Figure-1).The sender sends acknowledgement to it, which denotes the signature of the host, this in turn confirms the owner that the host is malicious or not.

1. Receive acknowledgment

2. Save the acknowledgment

4.4 Router:

Router keeps the track of the dynamically decided route. Sender sends the signature of the next host at each transition

1. Receive the signature of next host
2. Save it.

5. MODEL

The model shows the owner *H*, which launches the agent, the agent goes to host c_1 and then proceed further depending on the decision taken dynamically. Each host c_i, where $1 < i \leq n$ sends an acknowledgment to guard. A variant of this model is discussed in [17].

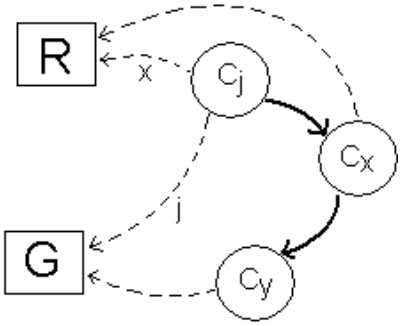

Figure 1. Model

6. DETECTION PROCEDURE

The owner waits for the result from the agent, when it finds that the time out has occurred it consults the Router and the Guard and for all the nodes listed in the Router checks for its acknowledgement with the Guard. In other words, it checks for an acknowledgment for c_i for all i listed in Router, when it finds that acknowledgment from c_i is missing it understands that c_i must be the malicious host and it resends the agent this time skipping c_i from the route and also after training the agent that c_i is malicious not too chose it dynamically.

Though there are several variations in the approach, it may be the case that c_i itself is malicious it does not want to respond or the acknowledgment might have lost. Sometimes the case may be that c_i is malicious and it corrupts the code and sends back the acknowledgment pretending to be a safe host, though this variation is tackled in our approach, as the guard checks for the code for each acknowledgment. In [2] it tackles the situation by taking into record the path history.

7. CPN MODEL

Colored Petri Nets (CPNs) is a language for the modeling and validation of systems in which concurrency, synchronization and communication play a major role [3]. It is a discrete-event modeling language combining Petrinets with the functional programming language Standard ML. It provides the graphical representation and the basic primitives for modeling concurrency, communication, and synchronization. Standard ML provides the primitives for definition of data types, describing data manipulation, and for creating compact and parametirsable models.

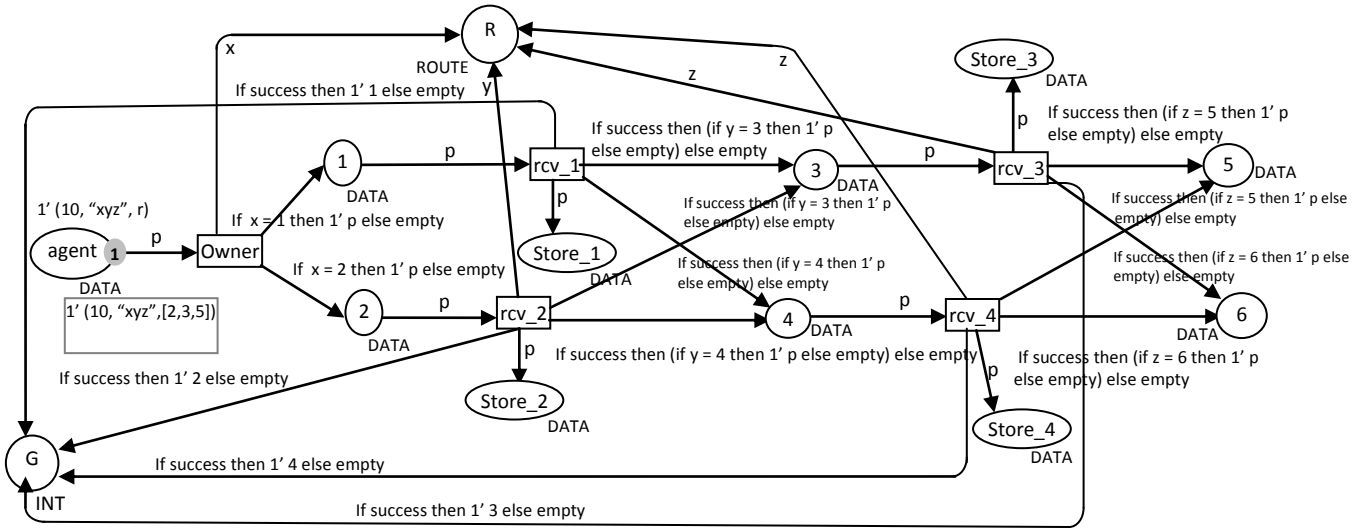

Figure 2. CPN Model

A CPN model is an executable model representing the states of the system and the events that can cause the system to change the state.

The proposed model shown in Fig. 2 has been simulated using CPNs. The CPN model has 13 places and 5 transitions. Owner of the agent is labeled as Owner. Senders/Receivers are labeled as RCV_1, RCV_2, RCV_3 and RCV_4.In between are intermediately places through which agent moves from one host to another .Copy of agent is saved in places called Stores. A place called Guard receives the entire acknowledgement labeled as 'G'; similarly a place called Router 'R' receives the entire route as per the algorithm discussed in section IV.

Declarations are used in CPN ML declaring the data types, constants, variables used in the model.

The agent is described in CPN ML by the following declarations:

colset id = int;

colse INT=int;

colset sc = string;

colset route =list INT with 1..5;

colset agent = product id*sc*route;

Screen shot of declaration is shown below:

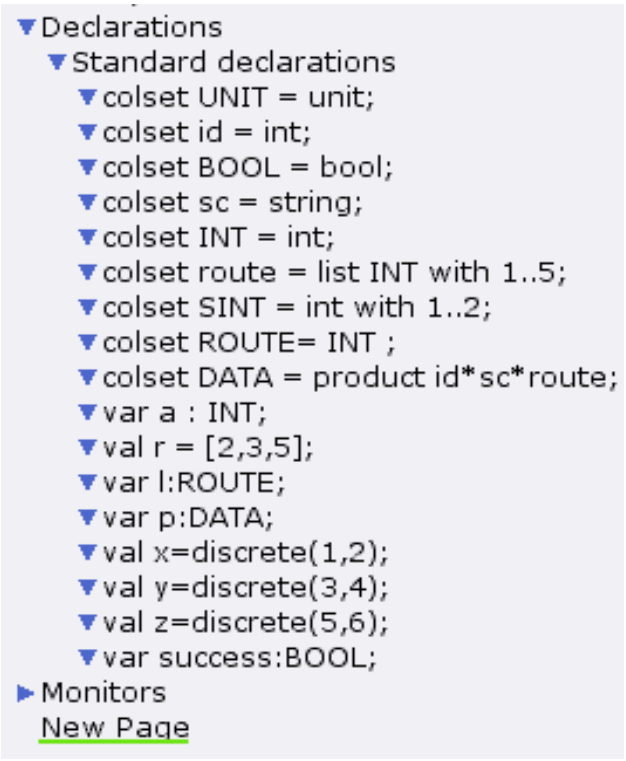

Figure 3. Screen Shot of Declaration

8. SIMULATION

Simulation is done 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. It provides a way to "walk through" a CPN model, investigating different scenarios in detail and checking whether the model works as expected. 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.

The Simulation of the model shows different markings. In the initial marking, agent is at the owner, there are two places where the agent can go depending on the dynamically decided route rcv_1 or rcv_2. On reaching rcv_1 or rcv_2 marking M1 is reached. In this marking the receiver rcv_1 or rcv_2 keeps a copy of agent, update the router with the next host and if 'success=true' sends the agent to next host (decided dynamically) and sends acknowledgment to guard. Marking M1, in which agent reaches to rcv_2 is shown in Figure 4.

Figure 4, shows that agent has reached to rcv_2, 'R' has been updated with the entry 1`2 which in CPN is read as 1 instance of value 2.

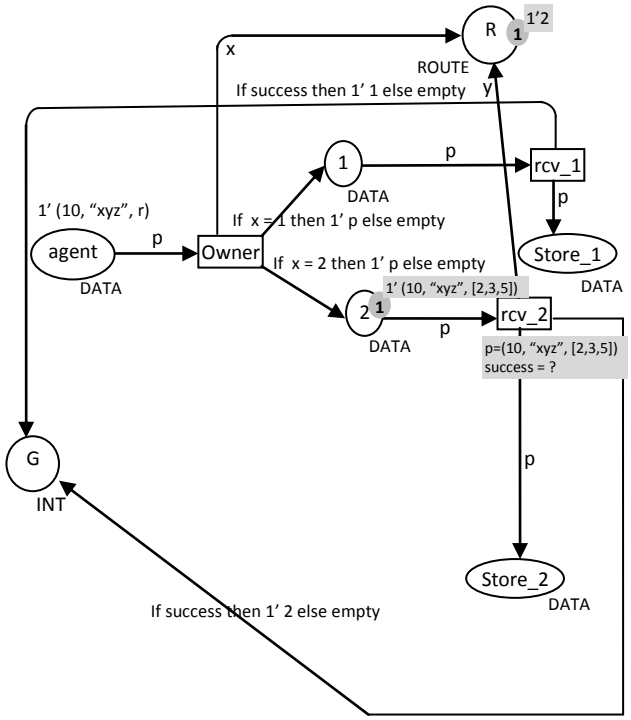

Figure 4. Marking M1.

Depending on the value of success in marking M1 there are two possibilities giving two variants of marking M2.If ' success = true' then only the agent is transferred to next host else it signifies that 'Denial Of Service' attack has occurred by host rcv_2.In M2, router receives the signature of next host from rcv_2.Figure 5, shows marking M2 for 'success=true'.

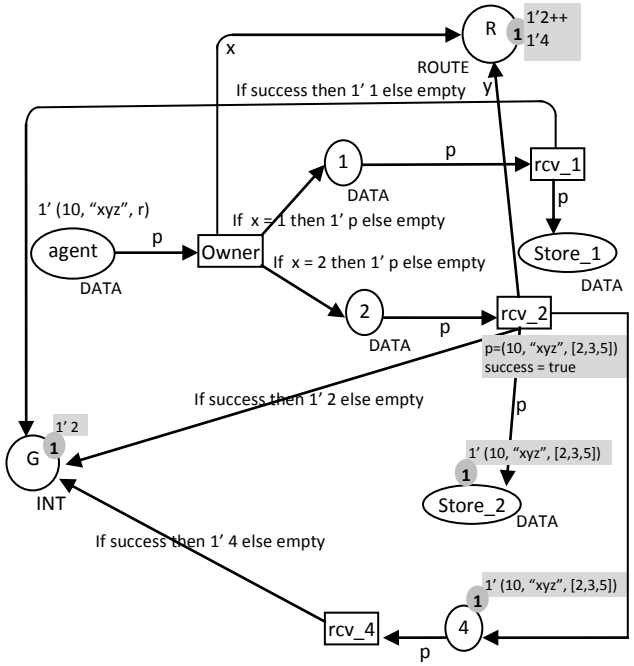

Figure5. Screen shot of Marking M2

As soon as the transition rcv_2 is executed for success= true, guard receives an acknowledgement from rcv_2 shown as 1`2 on the place labeled 'G' in Figure 5 and the agent moves to next host decided dynamically in above case rcv_4, so 'R' is updated with entry 1`4 shown as 1`2++1`4 signifying the complete route.(Agent moves from host 2 to host 4) upto rcv_4.

If Denial of Service attack occurs agent is not transferred further as shown in Figure 6.In this case Guard does not receive any acknowledgement but router is updated so owner can easily detect for malicious host when it finds missing acknowledgement with guard.

In Marking M2, for ' success=false' (Fig. 6) 'G' is empty and 'R' contains the entry 1`2++1`4 signifying that till now the route is from host 2 to host 4, but as 'G' is empty i.,e it has not received any acknowledgement host 2 is malicious it has not transferred the agent to next host i.e host 4. This shows that we can detect the malicious host whenever Denial of service attack occurs.

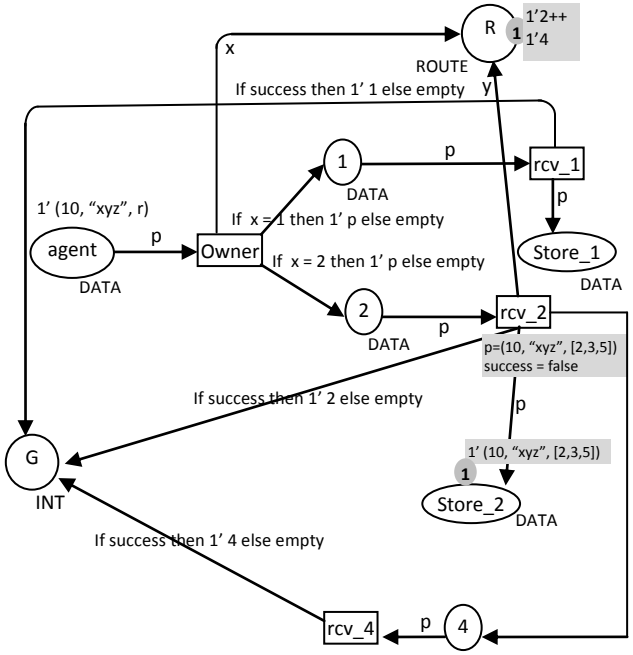

Figure 6. Screen shot of marking M2 for success = false

Similarly there can be a case in which host 4 is malicious, and host 2 has transferred the agent to host 2 correctly. This is the case if in Marking M2 'success=true' but rcv_4 is malicious which gives a new marking M3 in which the malicious host i.e rcv_4 is detected.(Fig. 7)

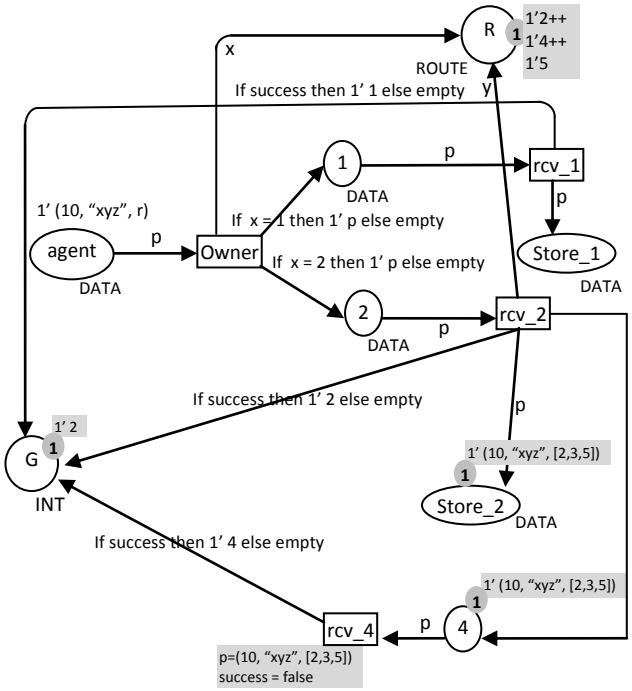

Figure 7. Screen Shot of Marking M3.

In marking M3 'success=false' agent has travelled till host 4 i.e rcv_4 but is lost now due to 'Denial of Service' attack by rcv_4.The above figure shows 'R' with 1`2++1`4++1`5 which signifies that the route is from 2 to 4 and from 4 to 5 but 'G' has only one acknowledgement i.e 1`2 which signifies that host 2 i.e rcv_2 has transferred the agent correctly but host 4 i.e rcv_4 is malicious.

Completely simulating the model several times makes different hosts malicious on different runs, and each time malicious host was detected by consulting the entries at 'R' and 'G'.

The simulation of the model proved that the malicious host for 'Denial of service' attack could be detected and thus 'Denial of service' attack could be prevented in the next journey by skipping the detected malicious host.

9. STATE SPACE ANALYSIS

Simulation can only be used to consider a finite number of executions of the model being analyzed. This makes simulations suited for detecting errors and for obtaining increased confidence in the correctness of the model. The situation of our model shows that Guard always has the correct acknowledgements and the malicious host can be detected but we can't ensure it to be 100% correct.

Full state space analysis represent all possible executions of the model being analyzed. The basic idea of full state spaces is to calculate all the reachable states (markings) and all state changes (occurring binding elements) of the CPN model and represent these in a directed graph where the nodes correspond to the set of reachable marking and the arcs correspond to occurring binding elements.

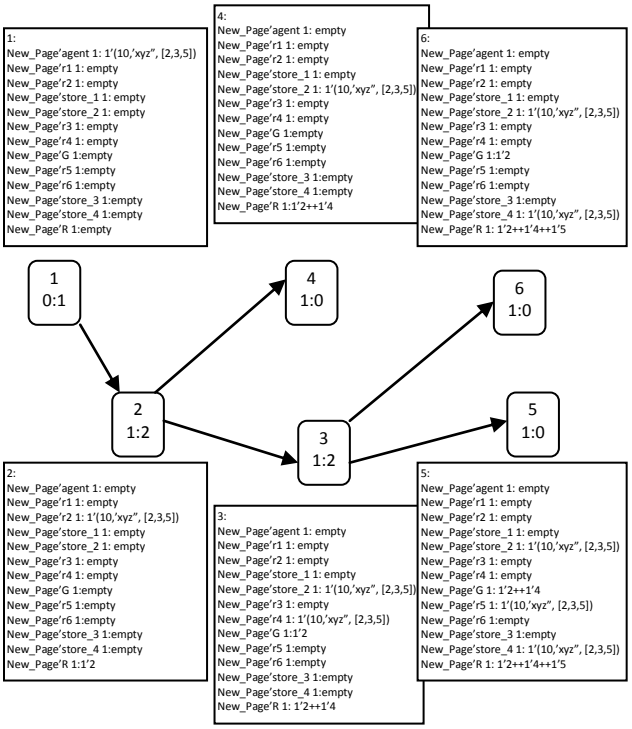

Figure 8. Complete state space analysis.

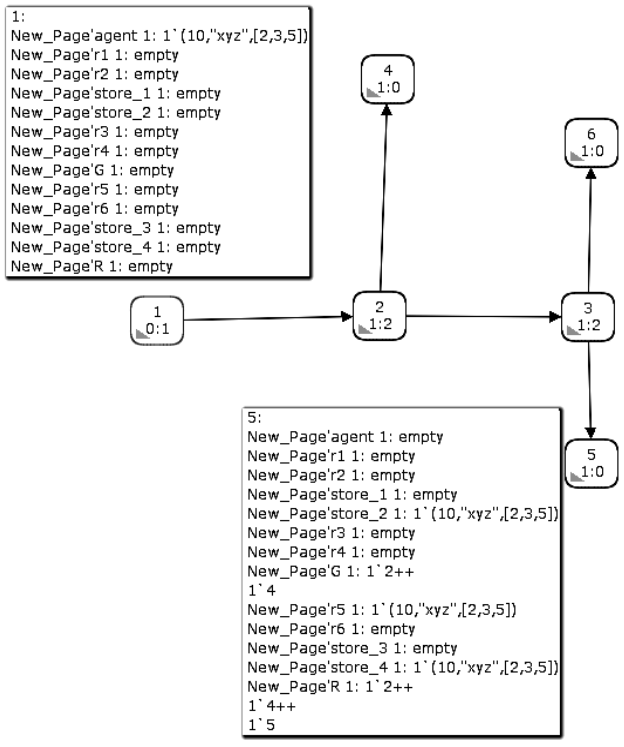

Figure 9. Part of State space analysis

State space diagram in Figure 9 shows G with marking 1`2++1`4 and R with marking 1`2++1`4++1`5 which is the case when no 'Denial of Service' attack has been done by any of the node in the dynamic route.

State space analysis also shows that by consulting the Guard and the Router owner can detect the malicious host and thus prevent 'Denial of service' attack in future journey by skipping that malicious host.

10. RELATED WORK

Many of the problems concerning the security of mobile agent systems, both protecting the host from malicious agents and protecting agents from malicious hosts, have been discussed in the literature. Counter measures for mobile agent security are well discussed in [18]. Execution tracing [9] is a technique for detecting unauthorized modifications of an agent through recording the agent's behavior during its execution on each host. In [10] Lofti and Samuel introduce the concept of mobile agent protection by clones. In [16] Sander and Tschudin introduce the concept of computing with encrypted functions and thus protecting the integrity and the privacy of the agent's computations.

Corradi present in [1] methods for protecting the agent's integrity —both making use of a Trusted Third Party and without it. In [6], Kim presented an adaptive migration strategy that can be used to avoid mobile agents from blocking or crashing.

This is achieved by a route reordering algorithm and a backward recovery algorithm.

In [4,5] Westhoffet describe methods for the protection of the agent's route against hosts spying out route information. One technique for ensuring that a mobile agent arrives safely at its destination is through the use of replication and voting [7]. The problem of detecting the black hole—a stationary process destroying visiting agents—in an anonymous ring is addressed in [15] visiting agents—in an anonymous ring is addressed in [15].

11. CONCLUSION AND FUTURE WORK

Simulation and State space analysis of the model shows that by consulting the Guard and the Router owner can detect the malicious host and thus prevent 'Denial of service' attack in future journey by skipping that malicious host.

The model discussed above is well suited for Dynamic mobile agent, but there may be certain case like sender sends the acknowledgement but corrupts the code, or the sender does not send the acknowledgement deliberately; such cases need to be considered.

12. REFERENCES

[1] Antonio Corradi, Marco Cremonini, Rebecca Montanari, and Cesare Stefanelli. Mobile agents integrity for electronic commerce applications. *Information Systems*, 24(6), 1999.

[2] Cao, Chun andLu, Jian 'Path-history-based access control for mobile agents', International Journal of Parallel, Emergent and Distributed Systems, vol 21: 3, pp 215 - 225, 2006.

[3] CPN Tools website: www.daimi.au.dk/CPNtools

[4] Dirk Westhoff, Markus Schneider, Claus Unger, and Firoz Kaderali. Methods for protecting a mobile agent's route. In *Information Security, Second International Workshop (ISW'99)*, number 1729 in LNCS. Springer Verlag, 1999.

[5] Dirk Westhoff, Markus Schneider, Claus Unger, and Firoz Kaderali. Protecting a mobile agent's route against collusions. In *Selected Areas in Cryptography, 6th Annual International Workshop (SAC'99)*, number 1758 in LNCS. Springer Verlag, 2000.

[6] Dong Chun Lee and Jeom Goo Kim. Adaptive migration strategy for mobile agents on internet. In *Technologies for E-Services (TES 2001), Second International Workshop, Proceedings*, number 2193 in LNCS. Springer Verlag, 2001.

[7] Fred B. Schneider. Towards fault-tolerant and secure agentry. In *Distributed Algorithms, 11th International Workshop (WDAG'97), Proceedings*, number 1320 in LNCS. Springer Verlag, 1997.

[8] Garrigues, C., et al. Promoting the development of secure mobile agent applications. J. Syst. Software (2009), doi:10.1016/ j.jss.2009.11.001

[9] Giovanni Vigna. Cryptographic traces for mobile agents. In G. Vigna, editor, *Mobile Agents and Security*, number 1419 in LNCS. Springer Verlag, 1998.

[10] Lotfi Benachenhou, *Samuel* Pierre, " Protection of a mobile agent with a reference clone," Elsevier , Computer Communications , vol 29, pp. 268-278, 2006.

[11] M.Aggarwal,Nupur,Pallavi, " Protecting Dynamic Mobile Agent against Denial of Service Attacks",AIP,Conference Proceedings,1324 (316),pp 316-318,2010.

[12] M. Schenider, B.Cubaleska "A method of protecting mobile agents against denial of service attacks" , Springer-Verlag Berlin Heidelberg , LNAI 2446, pp. 297–311, 2002.

[13] N.Desai,K.Garg,M.Mishra, Modelling Hierrarchical Mobile Agent Security Prorotocol Using CP Nets, Springer-Verlag Berlin Heidelberg, LNCS 4873, pp 637-649,2007.

[14] Price, Sean M. 'Host-Based Security Challenges and Controls: A Survey of Contemporary Research', Information Security Journal: A Global Perspective, vol 17: 4, pp 170 — 178, 2008.

[15] Stefan Dobrev, Paola Flocchini, Guiseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. In *Distributed Computing (DISC 2001), 15th International Conference, Proceedings*, number 2180 in LNCS. Springer Verlag, 2001.

[16] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In G. Vigna, editor, *Mobile Agents and Security*, number 1419 in LNCS. Springer Verlag, 1998.

[17] Venkatesan S, et al. "Advanced mobile agent security models for code integrity and malicious availability check.", J Network Comput Appl, doi:10.1016/j.jnca.2010.03.010 ,2010 .

[18] W.A.Jansen, "Countermeasures for mobile agent security," Elsevier, Computer Communications, vol. 23 , pp. 1667-1676 , 2000.