ISSN 1 746-7233, England, UK
World Journal of Modelling and Simulation
Vol. 5 (2009) No. 1, pp. 3-21

Petrinet based validation of novel location management techniques for mobile agents

Rama Sushil[1]*, Rama Bhargava[2], Kumkum Garg[3]

[1] SGRRITS Dehradun, India
[2] Department of Mathematics, IIT Roorkee, India
[3] Department of E & CE, IIT Roorkee, India

(*Received June 5 2008, Revised December 13 2008, Accepted January 6 2009*)

Abstract. Mobile agents are processes that can be dispatched from source computer and be transported to remote servers for execution, have been widely argued to be an important enabling technology for future systems. Location management is a necessity for locating mobile agents in a network of mobile agent hosts for controlling, monitoring and communication during processing and it still represents an open research issue. The cost of location management strategies mainly depends on the cost of search and update. We concentrated on reducing the cost of update and improving the speed of processing of the agents. We proposed two location management techniques; one is applicable for multi-region environment and other for single region environment. Technique for multi-region environment is named as Broadcasting with Search by Path Chase (BSPC), technique for single region is named as Multicasting and Movement based Location Management Technique (MMLMT). We used the tool TINA (TIme Petri net Analyzer) to model, analyze and to simulate both the techniques. It is found that the BSPC and MMLMT behave as expected and free from deadlock. We compared both the techniques with some existing location management techniques by evaluating the parameters-Location update fault rate, Interaction fault rate and Scalability.

Keywords: birth region, location finding protocol, location update and search cost, location management technique, mobile agents, mobile host, model analysis, Petri Net

1 Introduction

Mobile agents are software agents that can physically travel across a network to perform tasks on behalf of the user, under their own control as per their itinerary or deciding their movements dynamically according to execution results[19, 40]. Mobile agents are one form of mobile code[3, 17] that execute on machines that provide agent hosting. Mobile agent technology is being promoted as an emerging technology that makes it much easier to design, implement and maintain distributed systems and is an alternative to the client server model[7, 19]. There are many applications of mobile agents, including network management, e-commerce, distributed information retrieval, software distribution and load balancing. Mobile agent research has evolved with the creation of many different agent platforms. A mobile agent platform is a software that can create, interpret, execute, clone, transfer and terminate mobile agents e.g. Grasshopper[5], Aglets[2], ARCA[32], PMADE[26] etc. A mobile agent host is a node in the network which has a mobile agent platform installed in it.

A mobile agent location management strategy is required by an agent owner to control the agent for many reasons like to stop processing being done by the mobile agent, to give some more task to the agent, to know the intermediary results of the processing being done by the agent, to change its itinerary and for agent-agent communication and cooperation[11, 15, 20, 29, 35, 41]. Therefore, the ability to locate mobile agents while they are

* Corresponding author. Tel.:+91-135-2721763 fax: +91-135-625212. *E-mail address*: rama_sushil@hotmail.com.

migrating from one node of a network to another is of great importance in the development of agent-based applications, which is a part of the location management strategy. A major issue with location management is the high cost associated with location updates and search[6, 22–24, 37, 38]. The goal of an efficient location management strategy should be to minimize the combined cost of the location search and update. This cost is characterized by the time taken for each operation, number of messages sent, size of messages, or the distance the messages need to travel. Some problems which exist in location management protocols are unnecessary communication overhead, location data base server bottlenecks, high location update and search cost.

We concentrated for reducing the location update schemes so proposed two location management techniques. One for the multiregion environment, named BSPC (broadcasting with search-by-path-chase), which uses the general concepts of distributed systems[13, 28, 33, 34] and is based on search-by-path-chase (SPC)[36]. Other for single region, named as Multicasting and Movement based Location Management Technique (MMLMT). BSPC uses a novel location update scheme, combination of no update operation and path proxies, which costs less for applications having a low frequency of queries for contacting mobile agents i.e. low call to mobility ratio (CMR). MMLMT also uses a novel location update scheme, combination of no update operation and path proxies with optimal movements and is based on [23]. We model the BSPC and MMLMT using TIme PetriNet Analyzer (TINA)[2], perform the reachability analysis, structured analysis and simulated for total seven queries, with the status some in processing and some waiting to process, as an initial marking of the net.

The rest of this paper is organized as follows: section 2 briefs the related work emphasizing on location update schemes they use. Section 3 describes BSPC and MMLMT with their location update schemes and locating steps. Section 4 gives the PetriNet model of BSPC and MMLMT with their reachebility analysis and simulation results. Section 5 explains parametric evaluation and comparison for location management techniques- Path Proxies (PP), Data Logging (DL), Search By Path Chase (SPC), BSPC and MMLMT. We conclude the paper with a report on the possible extensions to BSPC and MMLMT in section 6.

2 Related work

Some previous studies have been made on the location management and message delivery protocols in a mobile agent computing environment: DL, PP, Blackboard, Shadow, SPC, HB, optimal location update scheme, MBLM, Scalable Hash-Based Mobile Agent Location Mechanism, Mailbox-based scheme for mobile agent communications. In Database logging protocol (DL)[35, 36] location information of mobile agents is stored at a specific server called location database server, upon every migration. The location information is used to find the mobile agents. As the location information is stored in centralized way, location server can represent a bottleneck in several conditions when number of mobile agents grows, mobile agents migrate frequently. If mobile agent moves far away from a specific server, the cost of location update comes to be relatively high. Moreover, if the location information in specific server is not latest means at the contact time if a mobile agent already left before catching the agent a following problem arises.

In path proxy protocol (PP)[35, 36] a mobile agent leaves a forwarding proxy at the source node at every migration. Mobile agent is located by following chain of proxies. No location update procedure is used. If path proxies are long and even one of the path proxies fails, mobile agent cannot be located. The shadow protocol[4, 20] is a combination of DL protocol and PP protocol. It provides functionality for locating agents, termination and for orphan detection. A mobile agent updates current location to an associated shadow according to TTL (time to live), which is a particular fix time interval. After the TTL a mobile agent updates its current location to its shadow instead of updating at every hop and by this method path proxies get cut short. In the shadow concept each application creates one or more dependency objects called shadows, a data structure on a place. An agent is an orphan when the shadow no longer exists.

In an optimal location update and searching algorithm for tracking mobile agent, mobile agent updates its location after it migrates through continuous d hosts since its last update. There is an optimal threshold d that makes the total cost of search and update minimized. If d assigned dynamically, it is called dynamic location update scheme[22, 31]. Blackboard protocol[39] maintains the blackboard at each node which is a shared information space for message exchange. When any mobile agent wants to deliver a message, it puts the

message in the blackboard no matter where receiving mobile agents are or when they read it. Afterword the receiving mobile agents move to the corresponding node where a message is stored and get the message. In this protocol, every node has storage where messages are deposited. In order to receive a message, mobile agents must move to the corresponding node, which makes unnecessary communication overhead occur. Because of not regular migration pattern message is not delivered immediately.

The SPC protocol[36] is a combination of DL protocol and PP protocol, applied to a multi-region mobile agent-computing environment. Location information is stored in a distributed way at a Region Agent Register (RAR) or a Site Agent Register (SAR). Agent is achieved by following a part of the links that the agent has left on two registers. Number of location update operations is reduced in SPC protocol by applying an optimal location update and searching algorithm for tracking mobile agent[22] on SPC. The modified protocol is named as MBLM (Movement Based Location Management)[6, 26]. This protocol is implemented on PMADE[38] and found to have lesser cost of location update than SPC protocol.

Scalable Hash-Based Mobile Agent Location Mechanism[21] have special agents called information agents (IAgents). IAgents are responsible for maintaining the current location of a set of agents. Set of mobile agents associated with each IAgents is determined through the hash function. This association changes over time as new IAgents are created or existing IAgents are merged depending on the system workload. A Home Blackboard (HB) protocol[14], location management and message delivery protocol, concentrated more for confirmed message delivery. A HB protocol is a hybrid of a DL protocol and a Blackboard protocol[39] in order to complement each other and overcome. Mailbox-based scheme[12] assigns each agent a mailbox to buffer messages, but decouples the agent and mailbox to let them reside at different hosts and migrate separately.

3 Location management technique for mobile agents

Location finding requires the presence of a suitable repository of the current locations of all the agents of the entire distributed system. Let us refer to a Name and Location Base (NLB) collecting the tuples (m, α, λ) with m the name of the agent α, which merely represent identifiers for the relative objects regardless of their real structure, or on the nature of the NLB (e.g., whether it is centralized or distributed, or where it is really located). Any location protocol for mobile agents deals with three aspects: name binding, migration, and location, each related to a particular phase in the agent's life. On NLB, we define four operations:
(1) Bind (m, α, λ): performed when a name m is assigned to mobile agent α, which is currently placed at location λ. This operation causes the insertion of the new tuple m, α, λ in the NLB. As the agent name has to be unique, this operation fails if a tuple with the name field equal to m already exists in the database.
(2) Newloc $(m, \alpha, \lambda^{'})$: performed when agent α changes its location, by migrating to $\lambda^{'}$. This operation changes the tuple (m, α, λ) already present in the NLB, into the new tuple $(m, \alpha, \lambda^{'})$.
(3) Find $m \rightarrow (m, \alpha, \lambda^{'})$: performed when agent α has to be located in order to interact with it. Given agent name m, this operation returns the relevant tuple if present in the NLB, thus determining the bound agent α and its current location λ.
(4) Unbind (m): performed when name m is no longer used (e.g., the agent is disposed of). This operation causes the deletion of the relative tuple from the NLB.

Location management for mobile agents implies finding a suitable way to implement the NLB and the four operations previously defined. Any solution has to deal with the traditional issues in distributed systems computing, meeting the reliability, efficiency, and scalability requirements. These aspects are tied to the real architecture of the NLB: As a centralized solution in a wide distributed environment is not reliable, efficient, or scalable, a distributed approach opens the question on how and where to distribute the information effectively. This affects the choice of the protocols used for executing the four primitives, which, if not suitably designed, can degrade the performance of the multiagent application. In this sense, the most critical operations are newloc and find, as they are used very frequently during the mobile agent's life. It may be the case that, once the current location of an agent's location is found but as that location is reached the agent has left the site, thus requiring repetition of the entire process (location finding + agent catching) until the agent is reached. Thus a mechanism is required to confirm the catching of the agent as it is found.

3.1 Location management with MMLMT

In this protocol mobile agent's current location database is distributed over all the nodes of the network. A node (on CLR-Current Location Register) in the network keeps the partial location information of its all the spawned agents and the agents passes through it. A CLR keeps the current location information with the fields; agent_id, cur_node_ip, ttl (time to live, it is a fix time interval which reduces gradually with a particular rate). As an agent is spawned insert function is invoked and relevant tuple is inserted in location database at the birth node. To reduce the locating time search process should start from corresponding birth nodes (A node is birth node for all the agents created on it) instead of from a particular node of the network always. MMLMT uses the naming scheme encoded with agent's birth node id. In our naming scheme, an agent name m has the following form:

$m =$ "agen" localname "@" BN_id, where BN_id is the name of the agent's node of birth and localname is the name of the agent chosen by the programmer. The unique name requirement imposes that there cannot be two or more regions with the same name, nor two or more agents, generated in the same region, with the same name. The uniqueness of the node name can be guaranteed by an authority, which assigns a name to a node when the latter is created (generally, this is a sporadic activity). In order to assure the uniqueness of localname, a binding protocol is needed to register the new agent and check that its name is unique in that region. The schema adopted presents a hierarchical structure with two levels, capable of a wide name space as compared to a flat-name scheme. Embedding the birth node name in m meets the requirement of providing a starting point for location finding.

For avoiding the unnecessary network load because of the orphan agents, we consider that the orphan detection and termination should be the important function of the location management technique. An agent is termed as the orphan if the application which generated the agent, is itself terminated. MMLMT identify orphan agent by the following method; As the mobile agent is dispatched a ttl is assigned to the agent (i.e. ttl is inserted in database). It reduces gradually with a particular rate. If in between or at the finishing of ttl mobile agent is not queried to be located and itself also not terminated, it is given another slot of ttl. Number of times a particular agent's ttl is allotted can be fixed say 10. Otherwise agent is supposed to be orphan and should be terminated and corresponding tuple must be deleted from the distributed database. Another stage for being treating as a orphan is when mobile agent is located and LMP is informing to the application (requested to locate an agent) for the agent location so that application can perform its communication with the agent, but it is found that application is already terminated. So in this situation also agent should be terminated and its all location information and proxies must be deleted from the network nodes.

For confirmed message delivery to the agent, a method is required to stop the agent from migration after its location at the moment location is found from the current location database. We implement it by locking the location tuple as the agent, s location is found, by this method mobile agent cannot update its location but to wait for tuple to be unlocked and consequently cannot migrate, till the agent is catched and communication is complete with the agent.

Location update procedure

Our update policy takes the advantage of mobile agent being in its immediate neighbor nodes (INN), by not update its location but only register at the destination node till it roams within INN, shown in Fig .1. Then at the first crossing out of the INN, it updates location at birth node location register and later updates after optimal number[23] of migrations agent leaves its proxies at the source nodes.

Locating steps

Every node in the network of mobile agent systems maintains a register of current location information. MMLMP functions are given below:

(1) Agent q makes a request to the location management protocol (MMLMP) to locate agent m. The $l.m.p.$ extracts the birth node of the agent to be located from its name.

(2) The birth node's CLR (Current Location register) is contacted. As per location information, the following steps take place

(2a) if current location field has value nil indicating agents is in any of its immediate neighbor node. Location Management Protocol multicasts a query to all its immediate neighbor hosts (NH). The host on which agent m is residing returns the 'agent found' message and locks m for migration, else

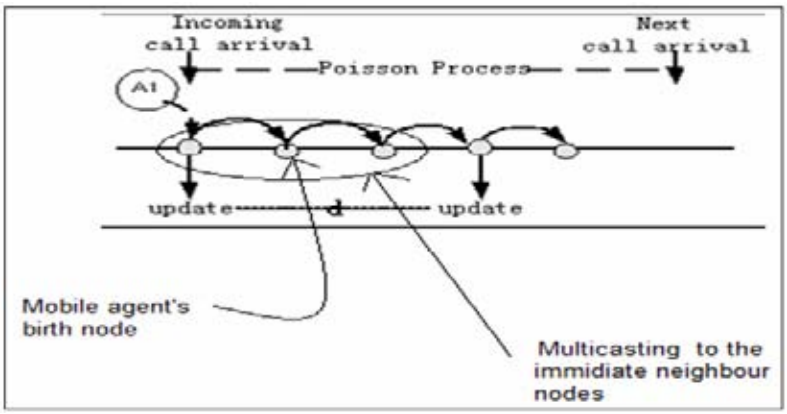

Fig. 1. Please write your figure caption here

(2b) the particular node's CLR is contacted and retrieves location information to start locating m (following path proxies) if not on that node.
(3) CLR returns the location information to the requesting host's MMLMT, which returns the location of m to agent q.
(4) Agent q communicates with agent m and informs agent m's ALT when completed.
(5) MMLMT then unlocks agent m, making it free to move.

3.2 Location management with BSPC

The protocol function is based on an efficient algorithm, which follows a part of the links left by the agents on the registers of the visited sites or regions and this protocol uses broadcasting to search an agent if mobile agent is roaming on the nodes of its birth region. Computing environment is considered as the collection of regions. A region is a collection of mobile agent hosts (Fig. 2). A particular mobile agent is spawned by a particular host, which belongs to a particular region called the birth-region of the agent. In each region there exists a site acting as a Region Agent Tracker (RAT), which manages a database called Region Agent Location Register (RALR). RALR stores the information about all the agents that have been created in the region or have transited through it. On each site there is a Site Agent Location Register (SALR), which contains information about all the agents that have transited through (in the past) or are at that location (see Tab. 1). Tab. 1 explains the meaning of the contents of the Location Agent Register.

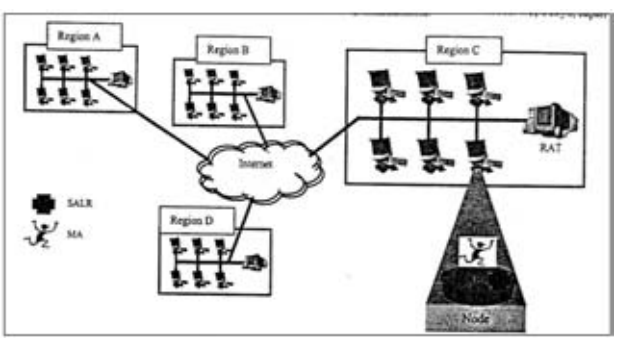

Fig. 2. Multi-region environment

Mobile agent migrations can be intraregional or interregional. Complete scenario of the BSPC protocol is shown in Fig. 3, as the agent migrates out of its birth region or roams within it. During location finding, each time a SALR is queried for an agent named m, an exclusive lock is set on the relevant tuple and is reset only if the agent does not reside in that place (i.e. the field of the tuple is nil), otherwise the lock is maintained.
Location update operations

Table 1. RALR/SALR tuple meaning

RALR Tuple	Meaning	SALR Tuple	Meaning
(m, GLI)	The agent is at location GLI or has traveled through it.	(m, nil, GLI)	The agent is at location GLI or has through it.
(m, GLI.region)	The agent is at region GLI or has traveled through it.	(m, nil, GLI.region)	The agent is at region GLI.region
		(m, α, nil)	The agent is in the same location as the SAR

Performance and reliability of the BSPC protocol strongly depends on the register update operations made during migration and binding operation. To avoid the burden of agent migration, the protocol aims to minimize interregional messages with respect to intraregional ones. With the assumption that the connections between sites in the same region are faster and more reliable than connections between different regions BSPC can have les overhead and improved efficiency. Commonly in WANs like Internet: Sites belonging to the same subnetwork6 are often connected by LANs (10-100 Mb/s), while connections between sub networks are point-to-point links working at a lower speed (64 Kb/s-2 Mb/s). The binding phase occurs when agent α is spawned, there is registration of the agent's name m and the birth location λ_m of α in $RALA_{m.region}$ ($m.region$ is the region of birth). This is handled by a two-step protocol performed by the platform executing at location $SALR_{\lambda_s}$. First, $RALR_{m.region}$ is contacted and, here, the tuple (m, λ_s) is registered. Then, the tuple (m, α, nil) is stored in $SALR_{\lambda_s}$. Before starting all the updating operations, an exclusive lock is placed on the entry of the SALR relevant to m, and is released when registration is finished (or an error occurs). The migration phase involves updating the location information of the migrating agent. Given λ_s and λ_d, the source and destination location, the sequence of operations can be split into two steps, performed in λ_s and λ_d respectively, before and after agent transfer. These steps vary according to whether λ_s and λ_d belong to the same region or not. In the case of intraregional migration $\lambda_{d.region} = \lambda_{d.region}$ and it is the birth region then the aim is to not to update the entries relevant to the migrating agent in both $SALR_{\lambda_s}$ and $RALR_{d.region}$. But the $RALR$ of the birth region is updated only when the agent is crossing the region.

If λ_s and λ_d belong to two different regions (interregional migration), the migration protocol has to update $SALR_{\lambda d}$, $SALR_{m.region}$ (if $\lambda_{d.region} \neq m.region$), by writing $\lambda_{d.region}$ as location information; it also updates by writing $RALR_{\lambda d.region}$ as location information and, finally, $SALR_{\lambda_s}$ registering the presence of the agent. This allows the location finding protocol, which starts from the region of birth of the agent, to reach the current region of the agent and, finally the current location. At location λ_s first the entry of the $SALR$ is locked, then, after agent transfer, the tuple $(m, nil, \lambda_{d.region})$ is stored on the $SALR$, then the tuple $(m, \lambda_{d.region})$ is stored on $SALR_{\lambda d.region}$ and, finally the lock is released. At the destination location λ_d when the agent transfer begins, a lock is placed on the entry of the $SALR$ and the $RALR_{\lambda d.region}$ is updated with λ_d as location information. When migration ends, first $SALR$ is released, and, finally the agent execution is resumed. At this time, if $\lambda_{d.region} \neq m.region$, a background (concurrent) process is started in λ_d that aims to remotely update $RALR_{m.region}$ by writing a tuple with $\lambda_{d.region}$ as location information. In the described protocol, $SALR$ locks play a fundamental role: They not only ensure exclusive access to $SALR$, but above all they help to resolve several inconsistencies which may happen in the case of concurrency between migration and interactions.

Locating steps

Broadcasting with Search-by-Path-Chase protocol (BSPC) functions are given below in algorithmic form.
(1) Agent q makes a request to the location management protocol ($l.\ m.\ p.$) to locate agent m. Location management protocol is available with each host as part of the mobile agent system or as a separate location management module. The $l.\ m.\ p.$ extracts the birth region of the agent to be located from its name.
(2) The birth region's RAT (Region Agent Tracker) is contacted. As per location information, the following steps take place
(a) if this region is the birth region, RAT broadcasts a query to all its member hosts (MH). The host on which

agent m is residing returns the 'agent found' message and locks m for migration, else
(b) the related RAT is contacted and the birth region RAT uses this information to start locating m in that region
(3) RAT returns the location information to the requesting host's $l.\ m.\ p.$ which then returns the location of m to agent q.
(4) Agent q communicates with agent m and informs agent m's birth region RAT when completed.
(5) RAT unlocks agent m, making it free to move.

4 Modeling and reachability analysis with TINA

A PetriNet is a mathematical formalism intended to be used for modeling, analysis and simulation of different kinds of distributed and parallel systems and processes[8, 16, 18, 25, 27]. In PetriNets, there are places and transitions; places are denoted by circles and are used to indicate system states like processing, accessing or waiting etc., transitions are denoted by directional edges and show the change of states after an event. Inhibition arcs are used for modeling error conditions in PetriNets. An inhibitor arc from a place to a transition disables the transition if the corresponding input place is not empty. In our petrinet model we used the inhibitor arc for locking function, when particular agent A_m is updating its location in the register that register should not be accessible for retrieving the location of A_m means register should be locked during that period of location update operation. A system that deadlocks is not correct in any situation so we check it by reachability

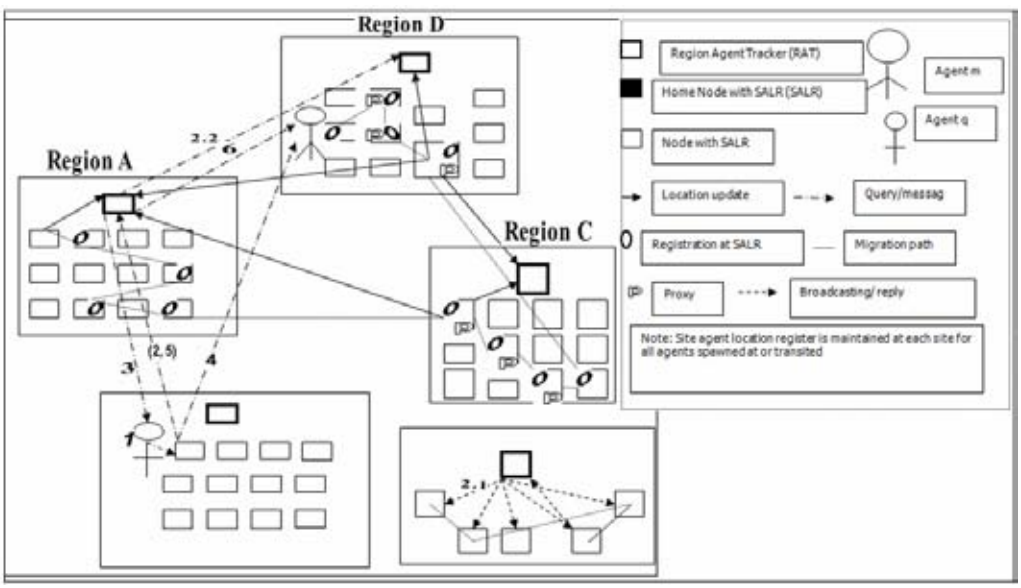

Fig. 3. BSPC tracking the agent with update operations and registration

analysis. In fact verifying whether a system exhibits deadlock is context independent. Validation is context dependent and can be done only with knowledge of the intended process. Enumeration Analysis[9] consists of the construction of an accessibility graph from the initial marking M0. A Petri Net is said to be live for an initial marking M0 if all the transitions are live. Places are allowed to contain several tokens. A token in a TPN may be in one of the two following states: available or unavailable. Initially each place p contains tokens available. A transition t may fire when, (1) There is at least one token in each of its input places, (2) There is no token in any of its inhibiting places, (3) Its enabling function evaluates to true, and (4) No other transition u with priority over t and satisfying (1), (2) and (3) exists. This removes the token from the input place and put the token in the output place. A token remains unavailable in input place during the transition occurs.

TIme petriNet Analyser software tool TINA[2, 10] facilitates the construction of a number of representations for the behaviour of Time Petri nets, in addition to the graphic-editing facilities. Various techniques are used to extract views of the behavior of nets, preserving certain classes of properties of their state spaces. For

Petri nets, these abstractions help prevent combinatorial explosion, relying on partial order techniques such as covering steps and/or persistent sets.

4.1 Modeling and analysis of MMLMT

We modeled the MMLMT taking different states as places of the location databases at different nodes of the network and transitions for different actions. A time petrinet analyzer software tool TINA is used to perform the automatic validation of the models by reachability analysis, which uses enumeration approach[9]. Reachability problem is decidable[16] although it takes exponential space (and time) to verify in the general case. A place Pi is said to be bounded for an initial marking M0 if for all marking accessible from M0 the number of tokens in Pi is finite. A Petri Net is said to be bounded for an initial marking M0 if all the places are bounded. Fig. 4 shows the model in graphical and text form. Places and transitions used in the model are described in Tab. 2 and Tab. 4. In reachability analysis MMLMT Petrinet is found bounded, has 168 markigs and 699 transitions and it takes 0.00s for this task. In liveness analysis it reports that net is live, has nil dead marking with nil transitions and 168 live markings with 10 live transitions, it takes 0.000s for this task. Similarly reachability analysis for level 2 and level 3 net is found bounded and live.

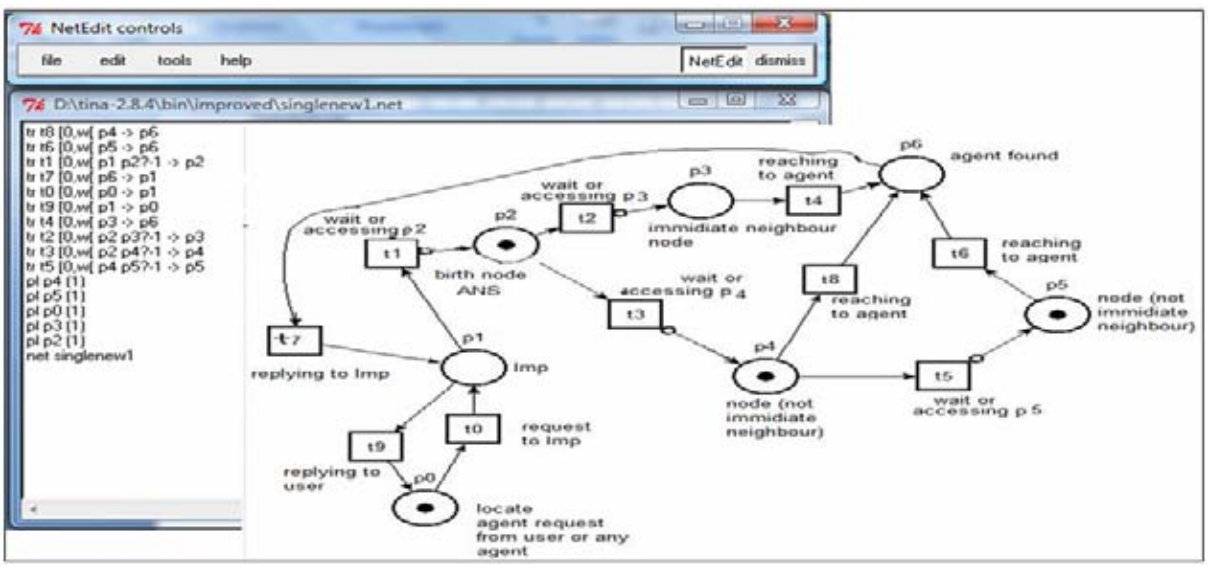

Fig. 4. Model of MMLMT

4.2 Modeling and analysis of BSPC

We have performed the reachability analysi[30] of the BSPC models for interregional migration and intraregional migration (when migration is in the region other than the birth region) shown in Fig. 6 and Fig. 8. We have performed the reachability analysis for finding the deadlock and liveness in BSPC models. For a clear description of the concurrency, conflicts and synchronization, we plan to consider only certain subsystems to model and analyze each using Time Petrinets. In the following discussion A_f is the agent who queries to find another agent A_m. Tab. 2 and Tab. 3 shows all possible places with their states and state transitions for the BSPC protocol in a TINA model, when locating a mobile agent during interregional migrations. We performed the reachability analysis for marking graph at level 1, level 2 and at level 3, structural analysis and stepper simulator too for models shown in Fig. 3 and Fig. 4. Marking graph at level 1, output in format quiet reports in Fig. 7; that petrinet has 13 plces, 23 transitions, TINA takes 0.0 seconds for this task. In reachability analysis, Petrinet (Fig. 7) is found bounded, has 33176 markigs, 240152 transitions and it takes 1.594s for this task. Liveness analysis in TINA reports that net is live, has nil dead markings and transitions, has 33176 live markings with 23 live transitions and takes 0.281s for this task. Similarly reachability analyses for level

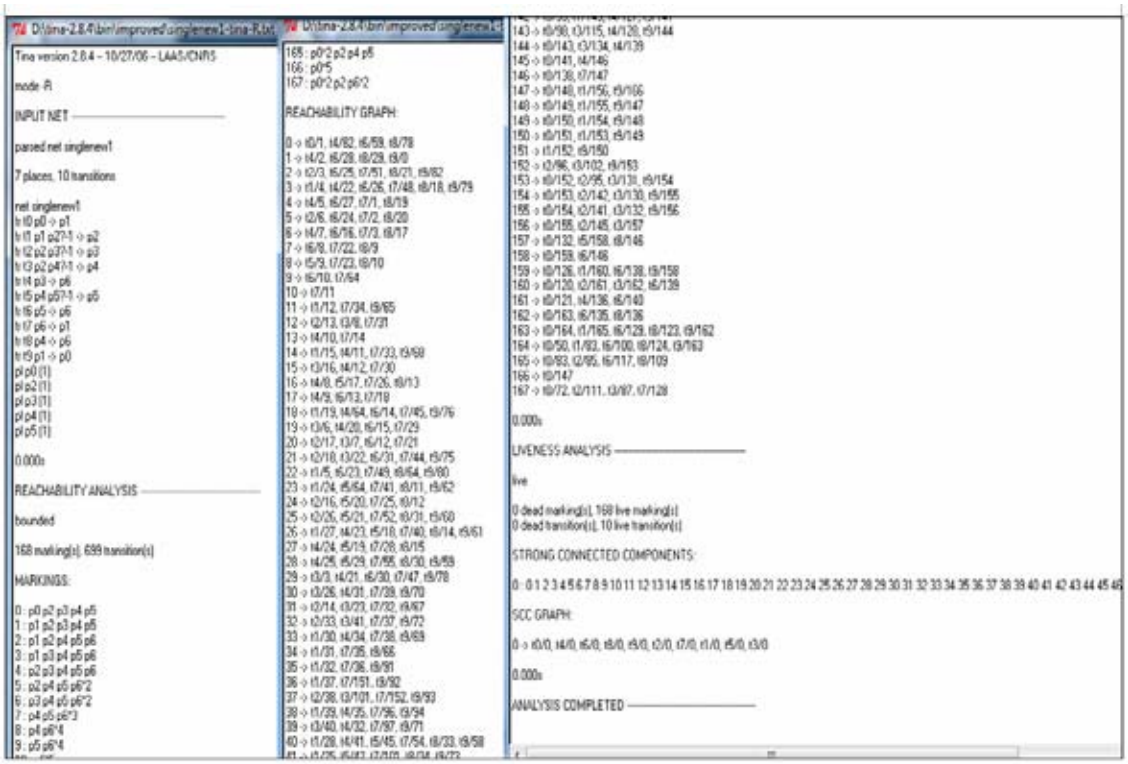

Fig. 5. Reachability analysis of the MMLMT model

Table 2. Places in MMLMP model

Places	Description
P0	agent A_f / user
P1	LMP
P2	Birth node's ANS
P3	Immidiate neighbour node
P4	Node1 different from immediate node
P5	Node2 different from immediate node and node1
P6	Agent found

Table 3. Transitions in MMLMP model

Transitions	Description
$t0$	location finding request to $P1$
$t1$	location finding request to $P2$
$t2$	waiting or accessing location information from $P3$
$t3$	waiting or accessing location information from $P4$
$t4$	Reaching to the agent
$t5$	waiting or accessing location information from $P5$
$t6$	Reaching to the agent
$t7$	Replying to LMP
$t8$	Reaching to the agent
$t9$	LMP sending result to user/ A_f

2 and level 3 of the petrinet is found bounded and live. In reachability analysis of the model in Fig. 8 is found bounded, has 182 markigs, 763 transitions and it takes 0.000s for this task. Liveness analysis in TINA reports that net is live, has nil dead markings and transitions, has 182 live markings with 11 live transitions and takes 0.000s for this task. Similarly reachability analyses for level 2 and level 3 of the petrinet is found bounded and live. For simulating the BSPC, Stepper simulator is run for model with initial marking shown in Fig. 8. There

is smooth running of all the queries in the form of tokens. With the shown initial markings and for some other initial markings we could see the processing of the seven querie successfully.

5 Parametric evaluation and comparison

We calculate the location update fault rate, interaction fault rate and scalability of DL, PP, SPC, BSPC and $MMLMT$ on the basis of parameter described in [36].

5.1 Location update fault rate

As far as migration is concerned, the fault rate can be expressed as the sum of the agent transfer fault rate and the location updating fault rate. The contribution of the first term does not depend on the location protocol used and will not be considered since it does not affect our comparison. For evaluating migration fault rate concentrating on location updating fault rate only. In the Database Logging (DL) technique, each migration implies the updating of a remote database and involves the network and a specialized site (the location database site):

$$R_f^{(DL)} = R_{fn} + R_{f\check{s}} \tag{1}$$

In the Path proxy (PP) technique, each migration involves the creation of a proxy in the local site:

$$R_f^{(PP)} = R_{fs} \tag{2}$$

In SPC protocol, during intraregional migration (birth region or other region), if the RAR of the region cannot be updated and only Site Agent Registers (SAR) are updated, the agent can still be located;

$$R_f^{(SPC)} = R_{fs} \tag{3}$$

While, for interregional migration the updating operations of the RAR of the source region and the agent's region of the birth can fail without affecting the correctness of the location phase. This implies that, in evaluating migration availability, we can consider the fault tolerance of the mandatory operations alone, unless the overall migration has had to fail because the agent could not be reached. These (operations without which locating mobile agent is not possible) are the updating of $SAR_{\lambda s}$, for intraregional migration, and updating of $SAR_{\lambda s}$ and $RAR_{\lambda d.region}$, for interregional migrations. By indicating the percentage of intraregional migration in the whole distributed system with x, we can express the total migration fault rate of the SPC protocol as a mean between intraregional fault rate (R_{fs}) and interregional fault rate ($R_{fs} + R_{f\check{s}} + R_{f\check{n}}$):

$$R_f^{(SPC)} = XR_{fs} + (1-X)(R_{fs} + R_{f\check{s}} + R_{f\check{n}}) = R_{fs} + (1-X)(R_{f\check{s}} + R_{f\check{n}}) \tag{4}$$

In case of BSPC protocol, during intraregional migration within birth region of the mobile agent, no update operation is required; agent can still be located (as within birth region broadcasting is used, on locating request only), means no RAR or SAR update is done. So intraregional fault rate:

$$R_f^{(BSPC)} = nil \tag{5}$$

During intraregional migration other than the birth region if RAR cannot be updated, still agent can be located (as SAR are updated). So intraregional fault rate:

$$R_f^{(BSPC)} = R_{fs} \tag{6}$$

While for interregional migration

$$R_f^{(BSPC)} = R_{fs} + R_{f\check{s}} + R_{f\check{n}} \tag{7}$$

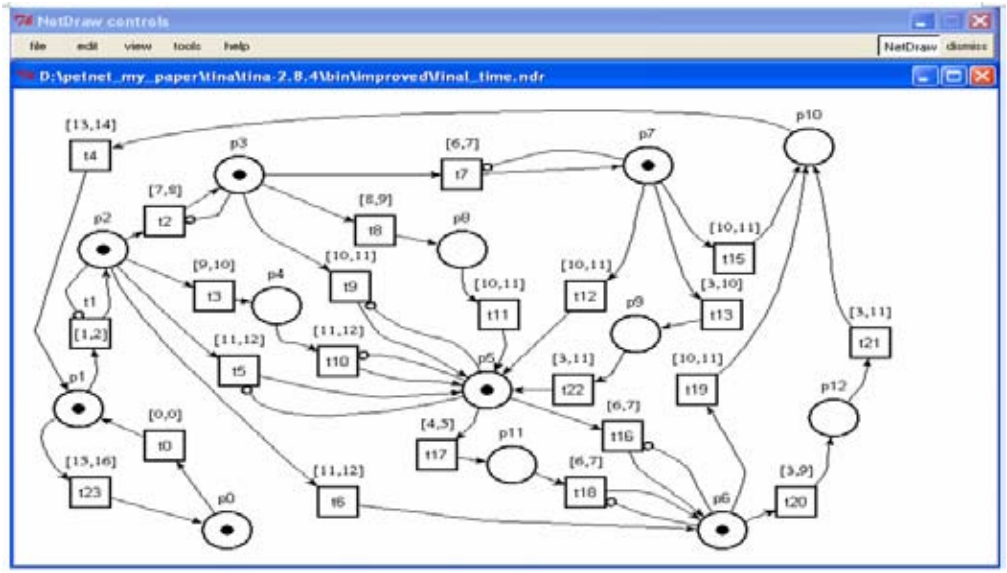

Fig. 6. Time petrinet model for interregional migration for BSPC

By indicating the percentage of intraregional (birth region) migration as $\dot{I}_b$, intraregional migration (other than birth region) as $\dot{I}$, Now we can express the total migration fault rate of BSPC protocol as a mean between intraregional fault rate R_{fs} and interregional fault rate $(R_{fs} + R_{f\check{s}} + R_{f\check{n}})$

$$R_f^{(BSPC)} = \dot{I}_b(nil) + \dot{I}R_{fs} + (1 - (\dot{I}_b + \dot{I}))(R_{fs} + R_{f\check{s}} + R_{f\check{n}})$$

or

$$R_f^{(BSPC)} = (1 - \dot{I}_b)(R_{fs} + R_{f\check{s}} + R_{f\check{n}}) - \dot{I}(R_{f\check{s}} + R_{f\check{n}}) \tag{8}$$

According to the location update scheme, each update operation is after optimal d migrations of the agent. So fault rate can be given by the following expressions:

$$R_f^{(MMLMT)} = R_{fs}/d \tag{9}$$

From Eq. (8), we observe that more is the value of $\dot{I}_b$ less is the total fault rate and $0 < \dot{I}_b < 1$ and $0 < \dot{I}_b < 1$. Using Eq. (1), (2), (4), (8) and (9), a graph shown in Fig. 9, is drawn to see the comparison of the location management techniques for location update fault rate. It shows that PP protocol offers the best location update availability degree i.e. least location update fault rate; DL has the maximum while BSPC offer better than SPC for more intraregional migrations within birth-region. MMLMT presents comparatively lower update fault rate for more values of the d and lowest update rate for optimal d.

5.2 Interaction fault rate

For interaction fault tolerance, we will evaluate the availability of the location finding protocol required at time t by generic agent wants to interact with the ith agent. Once again, this will be determined by calculating the total fault rate. In the DL technique, location finding involves a query to the location database site:

$$R_f^{(DL)'}(i,t) = R_{fn} + R_{f\check{s}} \tag{10}$$

Where $s(i,t)$ represents the number of migrations performed by the ith agent at time t. If PP is used to find an agent, the complete agent path starting from the home site has to be followed:

$$R_f^{(PP)'}(i,t) = s(i,t)(R_{fn} + R_{fs}) \tag{11}$$

Applying the query propagation technique, the interaction fault rate of the SPC protocol strongly depends on the updating operations performed during migration. The fault rate for SPC comprised between a minimum value $R_{f(best)}^{(SPC)'}(i,t)$ in best case when during migration all the updating operations are performed, and a maximum value $R_{f(worst)}^{(SPC)'}(i,t)$ in worst case when only the mandatory operations are performed.

Table 4. Description of Places in BSPC model

Places symbols	Meaning
$P0$	Initial state of agent A_f / user requesting to $l.\ m.\ p.$
$P1$	Initial state of agent $l.\ m.\ p.$ requesting to $RALR.$
$P2$	Initial state of the birth region $RALR.$
$P3$	Initial state of the source region $RALR_{\lambda s.region}$ ($P3$ fires when $RALR_{\lambda s.region}$ is not locked i.e., it is not engaged in the updating process by the same agent, to be located).
$P4$, $P11$, $P9$, $P8$, $P12$	Waiting states of A_f.
$P5$	Accessing destination region $RALR_{\lambda d.region}$
$P6$	Accessing $SALR_{\lambda d}$.
$P7$	Retrieving location information from $SALR_{\lambda s}$ where agent passed through.
$P10$	Agent found (final state)

Assuming that the searched agent is not in its birth region and in its itinerary it does not visit the same site and region more than once, then interaction fault rate for SPC in best case and worst case values are:

$$R_{f(best)}^{(SPC)'}(i,t) = 2(R_{fn} + R_{f\check{s}}) \tag{12}$$

$$R_{f(worst)}^{(SPC)'}(i,t) = r(i,t)(R_{fs} + R_{f\check{s}}) + s(i,t)(R_{f\check{n}} + R_{fs}) \tag{13}$$

The general case can be expressed as:

$$R_{f}^{(SPC)'}(i,t) = k_r(i,t)(R_{fn} + R_{f\check{s}}) + k_s(i,t)(R_{f\check{n}} + R_{fs}) \tag{14}$$

$K_r(i,t)$ represents the number of regions in the search path from the $RAT_{m.region}$ to the current location of the agent. it depends on the number of last consecutive interregional migrations featuring only the mandatory updating operations. $k_s(i,t)$ represents the number of locations (hosts) in the search path from the $RAR_{m.region}$ to the current location of the mobile agent, they depends on the number of last consecutive intraregional migrations featuring only the mandatory updating operations. Each time $RAR\lambda_{s.region}$ cannot updated, $k_s(i,t)$ increases as $RAR\lambda_{s.region}$ now belongs to the search path. While when migration completes with a success of all the remote register updating, $k_r(i,t)$ and $k_s(i,t)$ immediately reach their minimum values (respectively 2 and 0). So looking best case possibility, there is low probability that $k_r(i,t)$ and $k_s(i,t)$ can reach high values, unless there is very high fault rate. Obtaining an analytical expression for these two parameters requires a complex analysis, which, from our view does not give additional important information in determination of availability.

In BSPC protocol also, interaction fault rate strongly depends on the update operations performed during migrations in any application domain and same as in SPC, if we look upon the best application domain area for BSPC (in which there is a low frequency of locating request), like in a company which has several franchise spread all over the world, a single franchise is having its own LAN and each franchise's LAN is interconnected with each other. Most of the tasks are locally managed like pay slip generation of the employees. Each franchise has several departments. If the employees records are kept in distributed manner then particular mobile agent for collecting the record of the employees will at most roam in its birth region collect the record and prepare the pay slip. So most of the migrations will be intraregional and within the birth region. For normal application domain, with high mobility of mobile agents, with many intraregional migrations out of the birth region and there is high frequency of locating request.

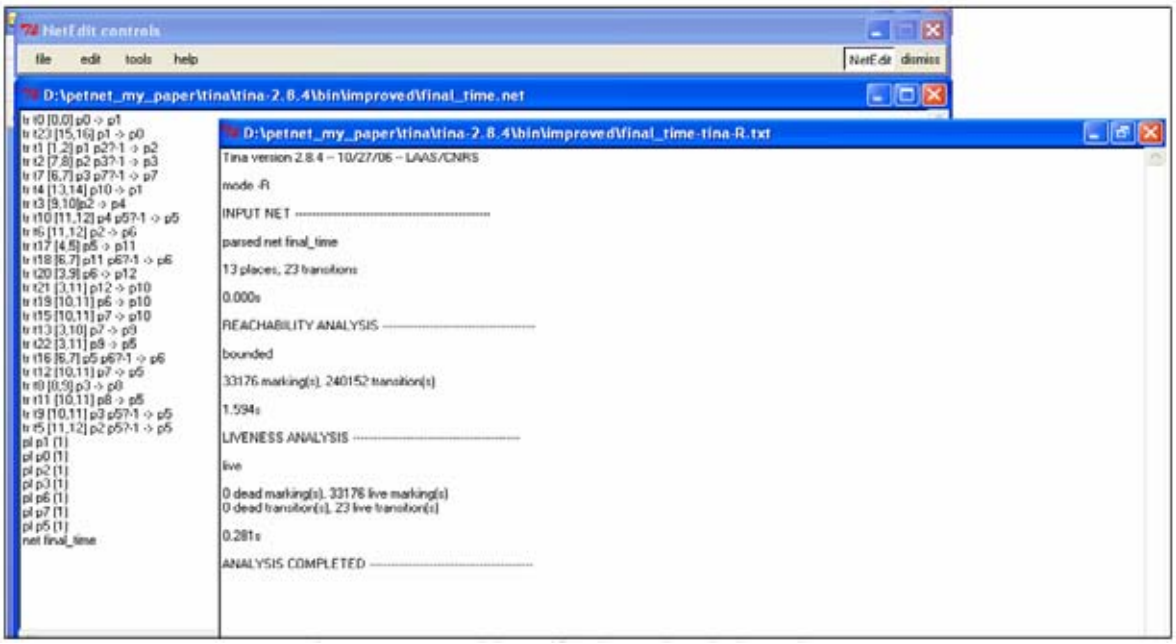

Fig. 7. BSPC model specification and analysis results

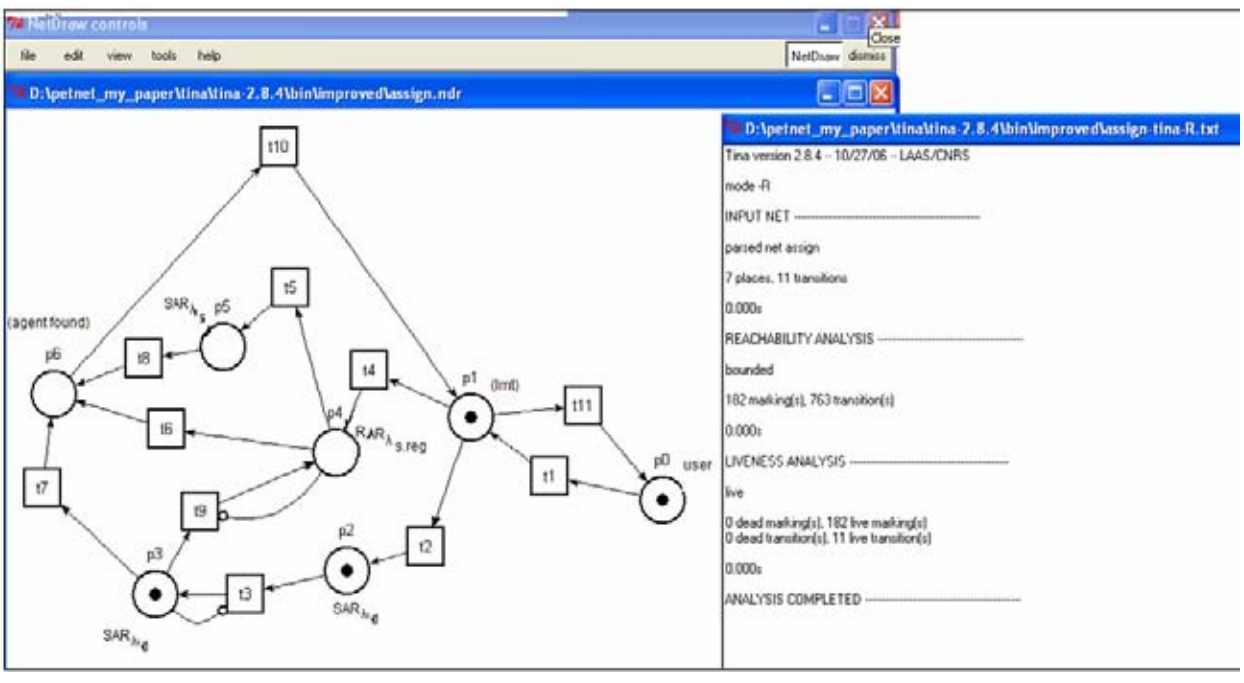

Fig. 8. Petri net model for intraregional migration in other than the birth region analysis result

$$R_{f(best)}^{(BSPC)'}(i,t) = 2(R_{fn} + R_{f\check{s}}) \tag{15}$$

For specific application domain area of low frequency of locating query and high degree of migrations within birth region

$$R_{f(best)}^{(BSPC)'} = R_{f\check{n}} + R_{f\check{s}} \tag{16}$$

$$R_{f(worst)}^{(BSPC)'}(i,t) = r(i,t)(R_{fn} + R_{f\check{s}}) + (1 - \dot{I}_b)s(i,t)(R_{f\check{n}} + R_{fs}) + \dot{I}_b s(i,t)(R_{f\check{n}} + R_{f\check{s}}) \tag{17}$$

The general case can be expressed as:

Table 5. Notaions used in the equations

R_{fn}	Fault rate of a generic network link
$R_{fn'}$	Fault rate of a generic network link belonging to a subnet or a LAN
R_{fs}	Fault rate of a generic site
$R_{fs'}$	Fault rate of a specialized site
$p^{'}$	Percentage of the intra-regional migration in the whole distributed system
$\lambda_1, \ldots, \lambda_n$	Locations of an itinerary of the mobile agent
$\sigma_1, \ldots, \sigma_n$	Regions in an itinerary of the mobile agent
$n(t)$	Total number of agents present at time t in the entire distributed environment
$s(i,t)$	Number of migrations performed by the ith agent at time t
$r(i,t)$	Number of regions crossed by the ith agent at time t
$T_{RAR}(\lambda_a, \lambda_b)$	Time required to remotely update, from location λ_a the RAR of the region λ_b
$T_{SAR}(\lambda)$	Time required to locally update SAR of the λ
$T_{QRAR}(\lambda_a, \lambda_b)$	Time required to perform a remote query from location λ_a to the RAR of region λ_b or to the loction database placed at λ_b
$T_{QSAR}(\lambda_a, \lambda_b)$	Time required to perform a remote query from location λ_a to the SAR of location λ_b
$\dot{I}$	Percentage of Intraregional migration (other than birth region)
$\dot{I}_b$	Percentage of Intraregional migration (within birth region)
P_k	Probability that the kth migration is of interregional type
$k_s(i,t)$	No. of hosts in the search path from RAR of the birth region $(RAR_{m.region})$ to the current location of the agent.
$k_r(i,t)$	No. of regions in the search path from RAR of the birth region $(RAR_{m.region})$ to the current location of the agent.

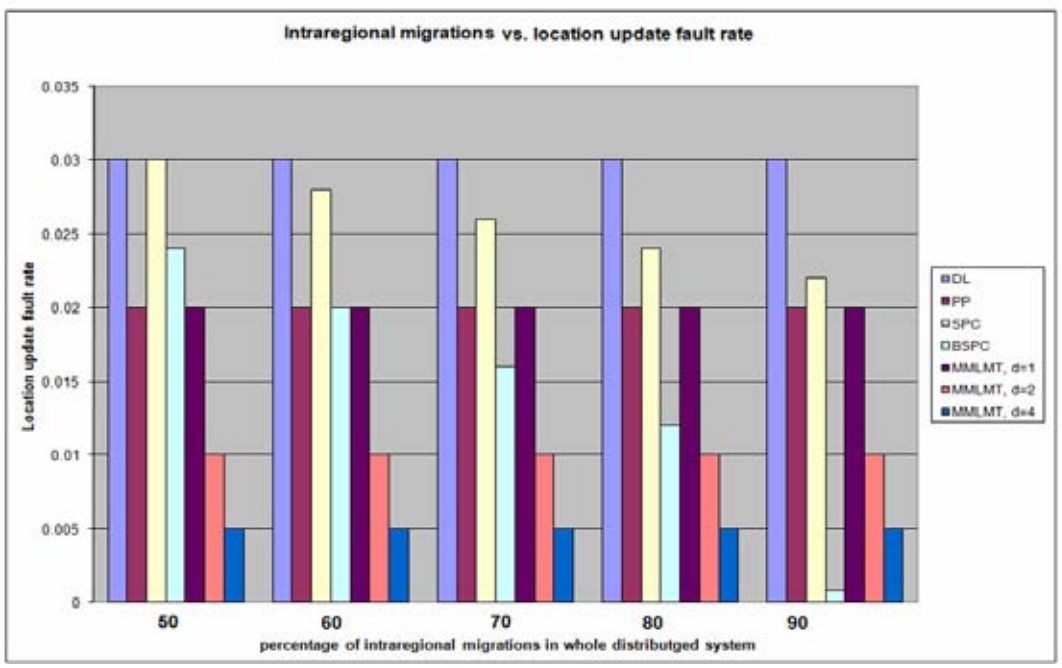

Fig. 9. Inraregional migrations vs location update fault rate

$$R_f^{(BSPC)^{'}}(i,t) = k_r(i,t)(R_{fn} + R_{f\check{s}}) + (1 - \dot{I}_b)k_s(i,t)(R_{f\check{n}} + R_{fs}) + \dot{I}_b k_s(i,t)(R_{f\check{n}} + R_{f\check{s}}) \quad (18)$$

$$R_f^{(MMLMT)^{'}}(i,t) = s(i,t)(R_{fn} + R_{fs})/d + k_n(R_{fn} + R_{f\check{s}}) \quad (19)$$

Where K_n is network topology dependent as is the number of immediate neighbor nodes. Now, we analyze the interaction availability with respect to the number of migrations made by the agent (hop count). The results are reported in the form of graph. Where SPC, BSPC and MMLMT are evaluated for some reference values of $K_r(i,t), k_s(i,t), s(i,t)$ and k_n. The graph in Fig. 10 shows that BSPC-best case in specific application domain offer the highest interaction availability, and also the BSPC-general case presents a very high value.

The worst performance is registered by the PP technique, which, for a high number of hop-count is also worse than the BSPC. For making these comparisons for the BSPC protocol, we have considered the case of searching of the agent always begin from Agent Name Server of the agent's birth region. MMLMT offers less interaction fault rate more value of d with same number of immediate neighbor nodes.

5.3 Scalability

Scalability can be evaluated by considering the overall distributed system response when the number of agents $n(t)$ and the number of migrations of each agent $s(i,t)$ increase. These parameters affect the network usage U_n. Network usage increases as the number of agents to locate grows. Site usage (U_s) indicates the number of entries used in all the location database of the distributed environment (including proxy elements). We express the global system usage (U) as the sum of U_n and U_s suitably weighted. The determination of the usage parameters may be hard, so a simplification of the system model is required. In determining an expression for U_n, the topology of the global network should be considered which leads to evaluating the usage of the links between the various sites (Site Agent Registers) and the specialized sites (like Region Agent Registers). However if we assume a uniform distribution of the sites over the various links, the overall network usage of each link can be expressed as proportional to a factor Tm/n, where Tm is the total number of messages and n is the total number of sites queried. In this case, network usage could play a substantial role if we consider the presence of a small number of location database sites n. If C is the average capacity each link then for the condition $Tm/n > C$. If we assume that n is adequate for the considered environment such that $Tm/n < C$, we can concentrate our analysis on the role of U_s. We calculate system usage by looking

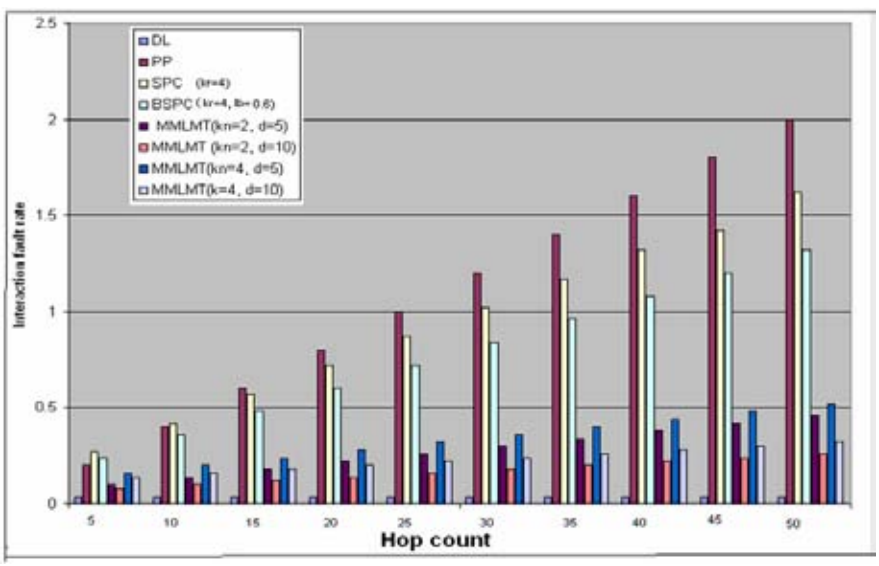

Fig. 10. Interaction fault rate vs mobile agent hop count

two factors first one is the number of entries related to the particular mobile agent at the sites (as the mobile agent register itself at the site) and second is the number of agents. For DL we assume the number of database sites such that network congestion is not caused. Sites involved in DL are the location database site and the current site of the agent where agent registers its presence. So

$$u^{(DL)}(t) = 2n(t) \tag{20}$$

For PP there is proxy on each site for reference to the next location of the agent, thus

$$u^{(PP)}(t) = \sum_{i=1}^{n(t)} (1 + s(i,t)) \tag{21}$$

For SPC, site usage can be evaluated by considering the registers growth in the worst case - when all the registers are updated during migration. In best case when only mandatory updating is performed;

$$u_w^{(SPC)}(t) = \sum_{i=1}^{n(t)} (s(i,t) + r(i,t)) \tag{22}$$

$$u_b^{(SPC)}(t) = \sum_{i=1}^{n(t)} (1 + r(i,t)) \tag{23}$$

In general global system usage can be expressed as;

$$u^{(SPC)}(t) = \sum_{i=1}^{n(t)} (1 + k_s(i,t) + r(i,t)) \tag{24}$$

For BSPC[27], consideration of intraregional migrations is the best case, as per its best application domain we assume that each region has adequate number of sites and mobile agents locating queries such that no network congestion occur when within the birth region broadcasting is used to locate the agent;

$$u_b^{(BSPC)}(t) = \sum_{i=1}^{n(t)} s(i,t) \tag{25}$$

For BSPC, worst case is when there are more inter-regional migrations and intra-regional migrations within the birth region are nil then it performs like SPC, so Eq. (20) is suitable for worst case of BSPC;

$$u_w^{(BSPC)}(t) = \sum_{i=1}^{n(t)} (s(i,t) + r(i,t)) \tag{26}$$

$$u^{(BSPC)}(t) = \sum_{i=1}^{n(t)} (k_s(i,t)) + r(i,t) \tag{27}$$

Now, if we consider that the database sites can be designed to handle a large number of entries, their contribution in evaluating scalability can be ignored. This leads to the following equations:

$$u^{(DL)'}(t) = n(t) \tag{28}$$

$$u^{(PP)'(t)} = \sum_{i=1}^{n(t)} (1 + s(i,t)) \tag{29}$$

$$u_b^{(SPC)'}(t) = n(t) \tag{30}$$

$$u_w^{(SPC)'}(t) = \sum_{i=1}^{n(t)} (s(i,t)) \tag{31}$$

$$u^{(SPC)'(t)} = \sum_{i=1}^{n(t)} (1 + k_s(i,t)) \tag{32}$$

$$u_b^{(BSPC)'(t)} = \sum_{i=1}^{n(t)} s(i,t) \tag{33}$$

$$u^{(BSPC)'}(t) = \sum_{i=1}^{n(t)} (k_s(i,t)) \tag{34}$$

$$u^{(MMLMT)'}(t) = \sum_{i=1}^{n(t)} \left(1 + \frac{s(i,t)}{d}\right) \tag{35}$$

We draw the graph shown in Fig. 11 using above relations for some reference values of $s(i,t)$, k_s and d. Fig. 11 shows that DL best scalability, PP shows the minimum scalability. In general BSPC scalability is better than SPC. For optimal d, MMLMT presents comparable more scalability.

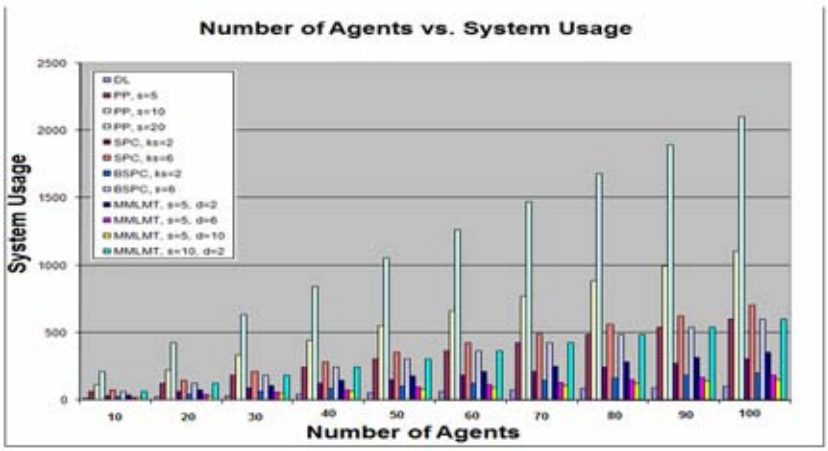

Fig. 11. Number of agents vs system usage

6 Discussion

We developed two location management protocols BSPC and MMLMT. BSPC applicable for multi-region environment and is designed for the applications having low frequency of locating queries (i.e. low call to mobility ratio) and with high frequency of migrations of the agents in their birth region as compared to any other region. Broadcasting is used to locate in birth region otherwise path of proxies are followed. Partially and limited use of broadcasting improves the speed of performing the task by the mobile agent. Inhibitor arcs were used for modeling lock function for the registers. As a result from every regular place, a transition is possible to the next regular place but if some error occurs (here error is basically the state when register is locked as being in use by the agent for location updating), transition cannot occur. MMLMT is designed for single region its update policy takes the advantage of mobile agent being in its immediate neighbor nodes by not update its location but only register at the destination node. Multicasting to the immediate neighbor nodes is used to locate the agents in their INN.

Verification, validation and simulation based on TINA are found to be an efficient way to improve the design process. Analysis results at different marking levels and in different output formats report that all the presented models of BSPC and MMLMT are bounded, live and deadlock free. Modelling and simulation successful results suggest that both the techniques can be implemented for any mobile agent platform. We calculated migration availability and interaction availability BSPC[5] in terms of fault rate of generic network link and generic site (0.02fault/haur) with $MTTF = 50$ hours and fault rate of generic network link belonging to a sub net (0.01fault/haur) with $MTTF = 100$ hours, results shown in Fig. 10 tells that interaction fault rate for BSPC is lower than SPC. PP protocol offers the best location updating availability degree; DL has the worst behavior while BSPC offers better than SPC for more intraregional migrations within birth-region. DL and SPC-best case present the best scalability, In general case BSPC scalability is better than SPC. MMLMT comparably presents low update, fault rate and system usage with optimal d, shown with different d.

BSPC protocol can be extended to use multicasting by RALR (called Region Agent Tracker also) for its own generated mobile agents only (whenever are they in the network) instead of broadcasting within birth-region. We intend to try for this extension in future in addition with BSPC and MMLMT implementation on Aglet network. Further we intend to calculate some more parameters like interaction overhead, migration overhead for DL, PP, SPC, BSPC and MMLMT.

References

[2] Http://www.trl.ibm.com/aglets.

[2] Http://www2.laas.fr/tina/distribution.php.

[3] S. M. amd W. LaForge, D. Chauhan. Mobile Objects and Agents (MOA). *Distributed System Eng.*, 1998, **5**(4).

[4] J. Baumann. *Control Algorithms for Mobile Agents*. Ph.D. Thesis, IPVR Stuttgart, 1999.

[5] C. Baumer, M. Breugst, et al. Grasshopper: a universal agent platform based on OMG MASIF and FIPA standards. *Technical report*, 2000. Http://citeseer.ist.psu.edu/baumer00grasshopper.html.

[6] M. Bavandla. *Improving the performance of location management protocols in a multiregion environment*. Master's Thesis, I. I. T. Roorkee, 2004.

[7] A. Berson. *Client/Server Architecture*, 2nd edn. McGrawHill Inc, USA, 1996. ISBN: 0-07-005664-1.

[8] B. Berthomieu, M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. **in:** *IEEE Transactions on Software Engineering*, *3*, vol. 17, 1991.

[9] B. Berthomieu, M. Menasche. A Enumerative Approach for Analyzing Time Petri Nets. **in:** *IFIP Congress 1983*, Paris, 1983. Http://www.laas.fr/tina/papers.php.

[10] B. Berthomieu, P. Ribet, F. Vernadat. The tool TINA-Construction of Abstract State Spaces for Petri Nets and Time Petri Nets. *International Journal of Production Research*, 2004, **42**(4). Http//www.laas.fr/ poribet/ PUBLICATIONS /ribet_2004_ijpr.ps.

[11] Bhattacharya, S. Das. LeZi-Update: An Information Theoretic Approach to Track Mobile Users in PCS Networks. **in:** *Fifth ACM/IEEE Ann. Conf. Mobile Computing and Networking*, 1999, 1–12.

[12] Cao, X. Feng, et al. Mailbox-based scheme for mobile agent communications. **in:** *IEEE Computer*, 2002, 5–60.

[13] S. Ceri, G. Pelagatti. *Distributed Database: Principles and Systems*. McGraw Hill, New York, 1984.

[14] S. Choi, M. Baik, C. Hwang. Location management & message delivery protocol in multi-region mobile agent computing environment. **in:** *the 24th International Conference on Distributed Computing Systems*, vol. 4, IEEE, 2004, 1063–6927.

[15] J. Desbiens, M. Lavoie, F. Renaud. Communication and tracking infrastructure of a mobile agent system. **in:** *31st Hawaii International Conference on System Sciences, Agent Mobility and Communication*, 1998, 54–63.

[16] J. Esparza, M. Nielsen. Decidability issues for petri nets- a survey. *Inform. Process Cybernet*, 1994, **30**: 143–160.

[17] Fuggetta, G. Picco, G. Vigna. Understanding code mobility. **in:** *IEEE Trans. Software Eng.*, *5*, vol. 24, 1998.

[18] K. Garg. *Design and performance validation techniques for distributed system using timed Petrinets*. Ph.D. Thesis, Imperial College of science and Technology, 1984.

[19] G. Harrison, D. Chess, A. Kershenbaum. Mobile agents: Are they a good idea? **in:** *Technical report*, IBM TJ Watson, 1995.

[20] JBaumann. A comparision of mechanisms for locating mobile agents. **in:** *faculty of computer science*, 1999, 1–25. Http://elib.uni stuttgart.de/opus/volltexte/1999/515/.

[21] Kastidou, E. Pitoura, G. Samaras. A scalable hash-based mobile agent location mechanism. **in:** *conference on distributed computing systems workshops (ICDCSW'03)*, 2003.

[22] J. Li, H. Kameda, K. Li. Optimal dynamic mobility management for pcs networks. **in:** *IEEE/ACM Trans. Networking*, *3*, vol. 8, 2000, 319–327.

[23] T. Li, J. Zhang. An Optimal Location Update and searching Algorithm for Tracking Mobile Agent. **in:** *AAMAS'02*, Bologna, Italy, 2002.

[24] B. Noy, I. Kessler, M. Sidi. Mobile users: To update or not to update? *ACM-baltzer J. Wireless Networks*, 1995, **1**(2): 175–186.

[25] Opsenica. Petri net based modeling and simulation of email alert system. **in:** *10th MELECON 2000 Conf.*, vol. 1, 2000, 49–52.

[26] R. Patel, K. Garg. Pmade- a platform for mobile agent distribution & execution. **in:** *the 7th International conference on information system analysis and synthesis*, vol. 4, USA, 2001, 287–293.

[27] J. Peterson. *Petri Net Theory and the Modeling of systems*. Prentice-Hall, Englewood Cliffs, 1981.

[28] E. Pitoura, I. Fudos. Distributed location databases for tracking highly mobile objects. *The Computer Journal*, 2001, **44**: 75–91.

[29] E. Pitoura, G. Samaras. Locating objects in mobile computing. **in:** *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, 2001, 571–592.

[30] Ramchandani. *Analysis of asynchronous concurrent systems with Timed Petrinets*. Ph.D. Thesis, MIT, 1973.

[31] V. Roth, J. Peters. A Scalable and Secure Global Tracking Service for Mobile Agents. **in:** *In MA'2001*, vol. 2240, 169-181, 2001, 169–181.

[32] C. Santoro. ARCA: A Framework for Mobile Agent Programming White Paper and Programmer's Tutorial. **in:** *Technical Report*, Univ. of Catania, 1998.

[33] J. Steen, H. Homburg, A. Tanenbaum. Locating objects in wide-area systems. **in:** *IEEE Communication Magazine*, 1998, 104–109.

[34] M. Steen, P. Homburg, A. Tanenbaum. Globe: A wide-area distributed system. **in:** *IEEE Concurrency*, 1999, 70–78.

[35] A. Stefano, L. L. Bello, C. Santoro. Naming and Locating Mobile Agents in an Internet Environment. **in:** *Third Intl. Conf. Enterprise Distributed Objects*, 1999.

[36] A. Stefano, C. Santoro. Locating mobile agents in wide distributed environment. **in:** *IEEE transactions on parallel and distributed systems*, *8*, vol. 13, 2002.

[37] D. Stefano, C. Santoro. The Coordination Infrastructure of the ARCA Framework. **in:** *Fourth Int'l ACM Conf. Autonomous Agents*, 2000.

[38] R. Sushil, K. Garg, R. Bhargava. Mobile agents: when to update their location? **in:** *Intl Conf. on Information & Communication Technology*, India, 2007, 852–857.

[39] X. Tao, J. Lul. Communication Mechanism in Mogent System. *Journal of Software*, 2000, **11**(8): 1060–1065.

[40] R. Tripathi, T. Ahamad, N. Karnik. experiences and future challenges in mobile agents programming. *Microsystems*, 2001, **25**(2): 121–129.

[41] P. Wojciechowski. Algorithms for location-independent communication between mobile agents. **in:** *Technical Report 2001/13*, Communication Systems Dept. and EPFL, 2001.