

USING π -CALCULUS FOR SPECIFICATION OF MOBILE AGENT COMMUNICATION

Gordan Jezic and Ignac Lovrek
University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Telecommunications
Unska 3, HR-10000 Zagreb, Croatia
{gordan.jezic, ignac.lovrek}@fer.hr

ABSTRACT

This paper presents formal specification and verification of agent migration and communication in a mobile agent network. The specification has been written in π -calculus process algebra based on link mobility and verified by Mobility Workbench analysis. The model consists of the mobile agents placed at distributed nodes and a mobility management agent responsible for message handling. The specification is focused on remote communication of the agents while migrating through the network. The system has been verified by Workbench model checking features.

KEY WORDS

Mobile agents, agent communication, formal specification, process algebra, verification.

1. Introduction

Multi-agent systems and especially the systems containing mobile and intelligent agents are promising paradigms for new generation networks that will offer full mobility of the users and services [1]. A mobile agent represents a user in a network that can migrate autonomously from node to node to perform some tasks on behalf of its user. Introduction of agent communication into the system reduces the processing time [2]. Communication of the agents allows co-operation and result-sharing between the agents operating simultaneously, and improves the overall system performance [3].

A typical communication protocol development process includes specification and design phase [4]. The bugs in final implementation appear either because of implementation or design errors. Implementation errors are easy to detect unlike design errors which are also expensive to correct as they require reverting to a design phase. It is, therefore, necessary to apply the techniques that detect errors at the early design phase. Some solutions to this problem are mathematically rigorous definitions in design. Such definitions are needed to specify the protocols and services, and to verify that the protocols fulfill their service [5, 6].

In a new generation network, where users and services are mobile, some entities change locations and connect to other different nodes at different points of time. Designing this kind of network where the entities are moving is more complicated than dealing with static environments. It is very important to formally define mobility of entities. Process algebras are a good choice for specification of concurrent and distributed systems. Over the past few years special attention has been paid to process algebras for defining mobile systems [7]. Calculi for specification of mobile processes are based either on mobility of the links or processes. The π -calculus, basic process algebra based on link mobility [8, 9], has been chosen in this paper as the language for formal specification of mobile processes implemented on the systems with mobile agents.

This paper elaborates on formal specification and verification of a mobile system called Mobile Agent Network (MAN) [2,10]. MAN consists of a multi-agent system where the agents co-operate and communicate in a network that allows agent mobility. The system is verified with automated analysis of Mobility Workbench [11].

The paper is organized as follows: A multi-agent system with agent communication and a model of a mobile agent network are described in Section 2. Architecture of the model, some suppositions, as well as agent mobility and communication in the model are defined in Section 3. Section 4 elaborates on formal specification of a mobile agent network in π -calculus process algebra. The system is verified with a model checking features of Mobility Workbench tool in Section 5. Section 6 concludes the paper.

2. Agent Communication in a Mobile Agent Network

Agent concepts and mobile software agents have become a part of the system and service architecture of the new generation networks. Agent paradigm is a promising choice for network-centric applications because it is intrinsically communication- and co-operation oriented. Application areas include the use of the agents in operation and management of networks, systems and

services. This is where agent's mobility offers important advantages because of the network load reduction, increased asynchrony between the communicating entities and higher concurrency.

Agent communication can be performed locally, at the same node, or remotely, between the agents placed at different nodes. Local agent communication is based on method invocation. Remote agent communication is based on message handling through the network. When an agent communicates with another one at a different node, it creates a message which is serialized, sent over the network, deserialized at a destination node and received by another agent.

The message sent consists of three parameters {sender, receiver, content}, where sender is the name of an agent sending the message/creating a new agent, receiver is the name of an agent receiving the message/a newly created agent, and content is the result sent/set as an input parameter for execution continuation. Each agent has a unique name that corresponds to communication address $name@hostname:port$, where name is a unique expression for an agent within the domain, hostname is the name (IP address) of the node S where an agent is placed, and port is a port number of a node to which the messages are being sent and received [2].

Performance of agent communication requires defining agent addressing, type of message exchange (synchronized or not) and message receipt (explicit or implicit). Agent addressing can be handled directly between the agents, or indirectly, using a centralized entity (according to FIPA standard) [12]. The problem in agent addressing is defining an agent location. One of the solutions is to write the agent path, so that each agent has its home and care-of address. Writing the path can be implemented using a home server, forwarding pointers or broadcast address [13].

Agent communication specified in the paper is based on a remote agent communication. It comprises modeling of a centralized agent responsible for agent addressing and synchronised message exchange with explicit message receipt.

2.1 Model Description

A mobile agent network is represented by the triple $\{A, S, N\}$, where A is a multi-agent system comprising co-operating and communicating mobile agents, S is a set of the processing nodes at which the agents perform services, and N is a network that connects the processing nodes and allows agent mobility. A multi-agent system, A , consists of n communicating mobile agents:

$$A = \{MA_1, \dots, PA_k, \dots, MA_n\}.$$

A multi-agent system resides in a network in which the agents perform the services requested by their users. It can be considered as a distributed system whose processing nodes communicate with each other over a communication network, N . A set of the processing nodes is denoted as follows:

$$S = \{S_1, \dots, S_i, S_j, \dots, S_{nc}\}.$$

Each node, S_i , is characterized by a set of services, $s(S_i)$, it provides. A mobile software agent, MA_k , is defined by $MA_k = \{name_k, address_k, service_k\}$, where $name_k$ is a unique agent identification and $address_k$ is its location and $service_k$ functionality it provides. For MA_k hosted by the node S_i or directed towards it $address_k = S_i$ and $service_k \in s_i$.

Network N provides message exchange between the agents placed at distributed nodes. In order to provide agent communication, the model introduces mobility management agent (MM) responsible for agent addressing and message routing. MM is a stationary agent that comprises database of agents location, and during communication receives the messages from a sender in order to forward them to the receiving agent. All agent communications are provided and messages pass through MM agent. The principle of communication is based on FIPA standard provided by Agent Management System, AMS [12].

The specification is focused on a remote agent communication while the agents are moving through the network. In that case, the specification is based on changing message direction from the old agent location to the new one, a new node. Message transfer between two nodes is specified with a link between them, so that communication handover is shown as moving of the link from MM agent towards a new node.

The model of mobile agent network consists of the following processes:

- mobile agents $MA_1, \dots, PA_k, \dots, MA_n$,
- network nodes $S_1, \dots, S_i, S_j, \dots, S_{nc}$, and
- stationary agents responsible for mobility management $MM_1 \dots MM_m$.

The number of stationary agents MM in MAN depends on the number of home networks in the system. MM agent is responsible for a set of nodes that represents one home network and contains data about the agents placed at these nodes. From MM point of view, agent communication can be considered in two parts: incoming call, from calling the agent who starts communication and sends the messages to MM agent, and outgoing call, from MM agent towards the called agent who receives the messages.

3. Definition of Agent Mobility and Communication

The model of MAN is represented in Figure 1. MAN is specified in the process algebra π -calculus. As a related work, the handover specification in GSM network can be found in [14].

MAN model consists of two mobile agents, MA_1 and MA_2 , one MM agent and four fully interconnected nodes. MA_1 and MA_2 are placed at different nodes (S_1 and S_2) and all communication and messages are exchanged over MM agent. MA_1 can migrate from S_1 to S_3 and back, and MA_2 from S_2 to S_4 and back.

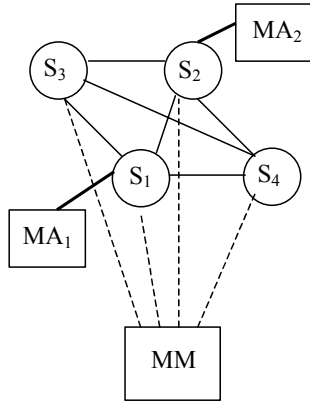


Fig. 1. MAN model

It is supposed that MA_1 is a calling agent who initiates communication, and that MA_2 is a called agent who receives the messages. During sending and receipt of the messages, the calling and called agents migrate from node to node, so that communication channel (link) has to be relocated (to a new node).

At the initial state communication has the following direction: $MA_1 - S_1 - MM - S_2 - MA_2$. If a called agent MA_2 has migrated to node S_4 , communication is directed towards $MA_1 - S_1 - MM - S_4 - MA_2$. Then, in the case of MA_1 migration, communication is routed towards $MA_1 - S_3 - MM - S_4 - MA_2$. Communication between the mobile agents can be divided into a calling part, $PA_1 - S_1 (S_3) - MM$, and a called part, $MM - S_2 (S_4) - PA_2$.

The model includes the input and output ports, *send* and *receive*. At *send* port new messages are generated. *Receive* port represents receipt of the messages. The system behaves as a buffer and all messages generated at *send* port should be received at *receive* port.

Figure 2 shows the initial state of the system. The link between the mobile agent and the node S represents position of the agent at that node. Initially, mobile agent MA_1 is placed at S_1 node, and MA_2 at S_2 . S_1 and S_2 nodes are denoted as active nodes (S_{1a}, S_{2a}), and S_3 and S_4 as passive nodes (S_{1p}, S_{2p}).

MM agent is divided into an active MM_a and a passive MM_p part (Figure 3). The active part is responsible for messages receipt and connection with the active nodes in communication. The passive part includes the links with the nodes where mobile agents are not momentarily placed.

The system consists of four processes:

- mobility management agent MM;
- active nodes S_{1a} and S_{2a} that participate in communication;
- passive nodes S_{3p} and S_{4p} that do not participate in communication, and
- mobile agents MA_1 and MA_2 .

During execution of the system, a calling agent MA_1 sends the data or updates the location message to MM.

During migration from S_1 to S_3 , MM agent must remove the existing connection (incoming handover procedure) to receive the messages at a new location and forward them to the called agent MA_2 . In the case of MA_2 migration, MM agent removes the link to a new node (outgoing handover procedure).

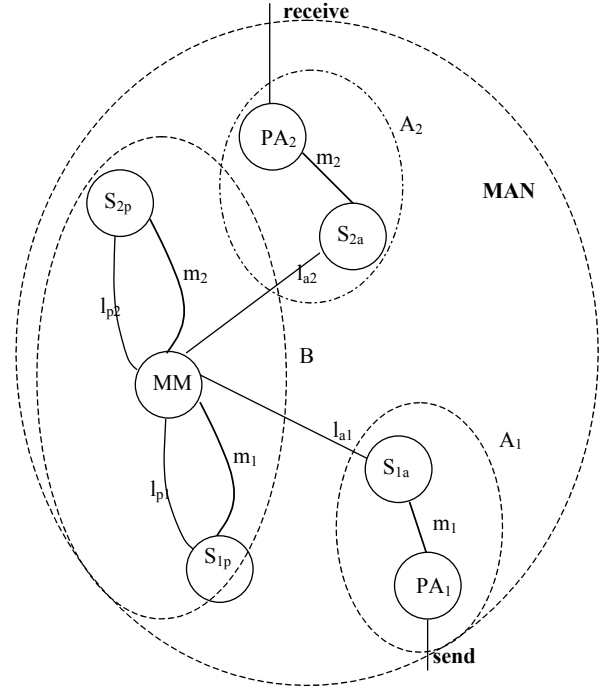


Fig. 2. The system at the initial state

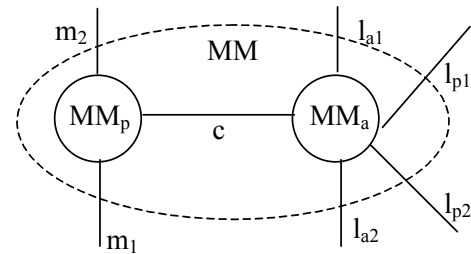


Fig. 3. MM agent

These procedures are specified formally in the π -calculus. Agent MM and passive nodes S_{1p} and S_{2p} form B part. Active node S_{1a} with mobile agent MA_1 forms A_1 part, and active node S_{2a} with MA_2 forms A_2 part. Links l_{a1} and l_{a2} connect MM with nodes S_{1a} and S_{2a} . Links l_{p1} and l_{p2} connect MM agent with passive nodes S_{1p} and S_{2p} . These links are fixed in the sense that their names cannot be sent as objects in communication and they are a part of the system.

Each node has a mobile link m which represents location of a mobile agent. If an agent is placed on the node, mobile link m connects these two processes. If a node does not have a mobile agent, mobile link m is connected with MM agent. Because of the existing two mobile agents (MA_1 and MA_2) placed at different nodes (S_{1a}, S_{2a}),

two mobile links connect the active nodes and mobile agents, and two links connect passive nodes (S_{1p} , S_{2p}) with MM agent.

4. Formal Description

The procedures of mobile agent communication and migration are the following. MA_1 sends messages *message* to MA_2 over the active part of MM, MM_a . MM_a receives the messages at link l_{a1} and forwards them to link l_{a2} towards node S_{2a} where destination agent MA_2 is placed. During communication each agent can start migration to another node. In that case a calling agent MA_1 sends an update location message, *move_request*. The system stops sending and receiving the messages and starts communication handover which is equal for the calling and the called agent. Handover starts when MM agent sends a *move_command* message to the mobile agent. Active part MM_a needs to receive a new channel m_{new} at link c . Two events are possible at this point: mobile agent can acknowledge the receipt of a *move_command* message by returning it or, if applicable, it can notify the error by sending *move_fail* message.

When MM receives a *move_complete* message the migration is completed and communication is directed to a new location. MM_a sends a *channel_release* message and the old channel is received at l_a and forwarded to link c as an archive. Active and passive links, l_a and l_p , exchange the roles and l_p is connected with the active, and l_a with the passive node.

If a *move_fail* message is received, MM_a returns a new link m_{new} to link c and goes to the initial state. In that case communication maintains the same direction and nodes do not exchange the roles. A decision as to whether agent migration and communication handover (*move_accept* or *move_fail*) are successful is modeled as a random event with identical possibilities.

Definition of a calling mobile agent is:

$$\overline{MA_1}(m_1) \stackrel{def}{=} \overline{send} \ v. \overline{ml} \ \mathbf{message}. \overline{ml} \ v. \overline{MA_1}(m_1) + \overline{ml} \ \mathbf{move_req}. \overline{MA_1}'(m_1)$$

$$\overline{MA_1}'(m_1) \stackrel{def}{=} m_1: [\mathbf{move_cmd} \Rightarrow m_1(m_{1new}). (\overline{ml} \ \mathbf{new} \ \mathbf{move_acc}. \overline{MA_1}(m_{1new}) + \overline{ml} \ \mathbf{move_fail}. \overline{MA_1}(m_1))].$$

Mobile agent MA_1 generates the messages (*send* action) and sends them (*message* action) to MM agent (m_1 link). At any time point a calling agent MA_1 can initiate migration by stopping message delivery and starting migration of a *move_request* message (*move_req* action) to MM. After sending the request, MA_1 agent waits for the response from MM at link m_1 (*move_cmd* action). If migration is successful, MA_1 returns a *move_accept* message (*move_acc* action), or in case of failure sends a *move_fail* message (*move_fail* action).

Active node S_{1a} receives the messages and forwards them to link l_1 towards MM agent. Besides data messages (*message*), S_{1a} can receive a *move_request* message from

MA_1 and forward the request to l_{a1} link towards MM agent. After that, it waits for the message to start migration (*move_cmd*) at l_1 link from MM. In that case S_{1a} forwards the message to mobile agent MA_1 and receives a message from MA_1 : either a *move_fail* message, if migration has failed or a *channel_release* message (*ch_rel* action), if migration has been successful. In the former case the old channel is released and a new one made active. Link m_1 is moved to a new location, the active node now being S_p .

$$\overline{S_{1a}}(l_{a1}, m_1) \stackrel{def}{=} m_1: [\mathbf{message} \Rightarrow m_1(v). \overline{la1} \ \mathbf{message}. \overline{la1} \ v. \overline{S_{1a}}(l_{a1}, m_1),$$

$$\mathbf{move_req} \Rightarrow m_1(v). \overline{la1} \ \mathbf{move_req}.$$

$$\overline{la1} \ v. \overline{S_{1a}}'(l_{a1}, m_1)$$

$$\overline{S_{1a}}'(l_{a1}, m_1) \stackrel{def}{=} l_{a1}: [\mathbf{move_cmd} \Rightarrow l_{a1}(v). \overline{ml} \ \mathbf{move_cmd}. \overline{ml} \ v. (l_{a1}[\mathbf{ch_rel}]. \overline{la1} \ m_1. \overline{S_{1p}}(l_{p1}, m_1) + m_1[\mathbf{move_fail}]. \overline{la1} \ \mathbf{move_fail}. \overline{S_{1a}}(l_{a1}, m_1))].$$

Passive node S_{1p} receives a *move_accept* message if an agent wants to migrate to it. If migration is successful S_{1p} sends a *move_complete* message (*move_com* action) to MM agent and S_{1p} node becomes active. In that case, S_{1a} and S_{1p} change their roles in the system.

$$\overline{S_{1p}}(l_{p1}, m_1) \stackrel{def}{=} m_1[\mathbf{move_acc}]. \overline{lp1} \ \mathbf{move_com}. \overline{S_{1a}}(l_{a1}, m_1).$$

MM agent consists of two parts: active part MM_a responsible for active links and nodes where the agents are placed (S_{1a} , S_{2a}), and MM_p part responsible for the nodes where the agents do not exist. MM_a agent forwards the messages received from the calling agent MA_1 (link l_{a1}) to link l_{a2} towards the called agent MA_2 . After receiving a *move_request* message from MA_1 , migration procedure starts, and MM_a agent sends a *move_command* message together with the new channel m_{1new} . Then MM_a agent receives either a message confirming successful performance of migration (*move_com* action) or a *move_fail* message informing about failed migration. Having received a *move_complete* message, MM_a agent releases the connection with the old node (*ch_rel* action), and active node becomes S_{1p} . MM_a agent also starts migration for the called agent MA_2 by sending a *move_command* message to link l_{a2} . The procedure for the called and calling agent is identical.

$$\overline{MM_a}(l_{a1}, l_{a2}, l_{p1}, l_{p2}, c) \stackrel{def}{=} l_{a1}: [\mathbf{message} \Rightarrow l_{a1}(v). \overline{in}(v). \overline{out} \ v. \overline{la2} \ v. \overline{MM_a}(l_{a1}, l_{a2}, l_{p1}, l_{p2}, c),$$

$$l_{a1}(\mathbf{move_req}). \overline{c}(m_{1new}). \overline{la1} \ \mathbf{move_cmd}. \overline{la1} \ m_{1new}. (l_{p1}[\mathbf{move_com}]. \overline{la1} \ \mathbf{ch_rel}. l_{a1}(m_{1old}). \overline{c} \ m_{1old}. \overline{MM_a}(l_{a1}, l_{a2}, l_{p1},$$

$$\begin{aligned}
& l_{p2}, c) + l_{a1}[\mathbf{move_fail}]. \bar{c} m_{1old}. \\
& MM_a(l_{a1}, l_{a2}, l_{p1}, l_{p2}, c), \\
& c(m_{2new}). \bar{la2} \mathbf{move_cmd}. \\
& \bar{la1} m_{new}. (l_{p2}[\mathbf{move_com}]. \bar{la2} \mathbf{ch_rel}. l_a \\
& 2(m_{2old}). \bar{c} m_{2old}. MM_a(l_{a1}, l_{a2}, l_{p1}, l_{p2}, c) \\
& + l_{a2}[\mathbf{move_fail}]. \bar{c} m_{2old}. MM_a(l_{a1}, l_{a2}, \\
& l_{p1}, l_{p2}, c) \\
MM_p(c, m_1, m_2) & \stackrel{def}{=} \bar{c} m_1. c(m_1). MM_p(c, m_1, m_2) + \bar{c} m_2. \\
& c(m_2). MM_p(c, m_1, m_2) \\
MM(l_{a1}, l_{a2}, l_{p1}, l_{p2}, m_1, m_2) & \equiv (c)(MM_p(c, m_1, m_2) | \\
MM_a(l_{a1}, l_{a2}, l_{p1}, l_{p2}, c)
\end{aligned}$$

Active and passive nodes on the called side, S_{a2} and S_{p2} , have the same role as on the calling side, and the specification is not additionally explained.

$$\begin{aligned}
S_{2a}(l_{a2}, m_2) & \stackrel{def}{=} \bar{l}_{a2}[\mathbf{message} \Rightarrow l_{a2}(v). \\
& \bar{m2} \mathbf{message}. m 2v. S_{2a}(l_{a2}, m_2), \\
& \mathbf{move_cmd} \Rightarrow l_{a2} \\
& (v). \bar{m} 2\mathbf{move_cmd}. \bar{m} 2v. \\
& (l_{a2}[\mathbf{ch_rel}]. \bar{la2} m. S_p(l_{p2}, m_2) \\
& + m[\mathbf{move_fail}]. \bar{la2} \mathbf{move_fail}. S_a(l_{a2}, \\
& m_2))] \\
S_{2p}(l_{p2}, m_2) & \stackrel{def}{=} m[\mathbf{move_acc}]. \bar{lp2} \mathbf{move_com}. S_{2a}(l_{a2}, m_2).
\end{aligned}$$

The called agent MA_2 either receives the messages (*receive* action) from the calling agent MA_1 or migrates to a new node. In the case of migration MA_2 receives a *move_command* and sends back either a *move_accept* or a *move_fail* message.

$$\begin{aligned}
MA_2(m_2) & \stackrel{def}{=} m: [\mathbf{message} \Rightarrow \bar{m}(v). \text{receive}(v). MA_2(m_2) \\
& \mathbf{move_cmd} \Rightarrow m(m_{new}). (\bar{mnew} \mathbf{move_acc}. MA_2(m_{new}) + \\
& \bar{m} \mathbf{move_fail}. MA_2(m_2))]
\end{aligned}$$

Formal specification of MAN system (System) consists of a parallel composition of defined processes.

$$\begin{aligned}
B(l_{a1}, l_{a2}, l_{p1}, l_{p2}) & \equiv (m_1)(m_2)(MM_a(l_{a1}, l_{a2}, l_{p1}, l_{p2}, c) | \\
& S_{1p}(l_{p1}, m_1) | S_{2p}(l_{p2}, m_2)) \\
A_1(l_{a1}) & \equiv (m_1)(S_{1a}(l_{a1}, m_1) | MA_1(m_1)) \\
A_2(l_{a2}) & \equiv (m_2)(S_{2a}(l_{a2}, m_2) | MA_2(m_2)) \\
System(l_{a1}, l_{a2}, l_{p1}, l_{p2}) & \equiv (B(l_{a1}, l_{a2}, l_{p1}, l_{p2}) | A_1(l_{a1}) | \\
& A_2(l_{a2}))
\end{aligned}$$

5. Automated Verification

Automated verification of the system is done by Mobility Workbench tool (MWB) [11]. MWB is capable of designing and verification of the mobile concurrent systems defined in the π -calculus process algebra. An

example of the automated analysis of a handover protocol for a mobile telephone system can be found in [15].

Verification of MAN system comprises the incoming and outgoing part of agent communication. It is needed to observe the system from outside viewpoint, so external actions are taken into consideration. For the incoming part of communication the mobile agent generates messages (external action *send*) towards the mobility management agent (*in* action). In the outgoing part MM agent forwards the messages (*out* action) and the called agent receives them (external action *receive*).

The system is defined as a buffer which generates messages at port *send* (calling agent) and forwards them through MM agent (*in*, *out*) to port *receive* (called agent). Each process of the system (mobile agent MA, mobility management agent MM and node S) includes at most one message in one time interval, so that calling and called parts can include maximum three messages each. Agent migration is specified as a random event, so both successful and unsuccessful migrations have the same possibility.

Agent migration and communication handover are presented as an internal action for the system. With the start of a calling agent migration, all messages in the system must be delivered to MM agent on port *in* before a new message is generated and sent to MM.

Based on a defined proposal, the expected behaviour of the calling part of communication is the following:

$$\begin{aligned}
MAN_{10} & \stackrel{def}{=} \bar{send} v. MAN_{11}(v) + \text{move}(\text{req}). \tau. MAN_{10} \\
MAN_{11}(v_1) & \stackrel{def}{=} \bar{send} v. MAN_{12}(v_1, v) + \text{in}(v_1). MAN_{10} + \\
& \text{move}(\text{req}). \tau. \text{in}(v_1). MAN_{10} \\
MAN_{12}(v_1, v_2) & \stackrel{def}{=} \bar{send} v. MAN_{13}(v_1, v_2, v) + \text{in}(v_1). \\
& MAN_{11}(v_2) + \text{move}(\text{req}). \tau. \text{in}(v_1). \text{in}(v_2). MAN_{10} \\
MAN_{13}(v_1, v_2, v_3) & \stackrel{def}{=} \text{in}(v_1). MAN_{12}(v_2, v_3)
\end{aligned}$$

MAN_j for $j = \{0,1,2,3\}$ corresponds to the buffer which includes j messages. The messages are the parameters v_1 , v_2 and v_3 . Each MAN_j where $j < 3$ goes to MAN_{j+1} , generating a new message at the *send* port. For each MAN_j , $j > 0$ there is MAN_{j-1} after receiving the message at *in* port. For each MAN_j , $j < 3$ after performing the internal action τ and action *in*, the system goes to the initial state.

The calling part of MAN system includes three one-side buffers. At the initial state the system is empty (MAN_{10}). MA_1 can start either generating and sending the messages (*send*) or a migration procedure (*move*). If sending messages, the system accepts the first message and goes to state MAN_{11} in which it includes exactly one message. If starting migration procedure, the system goes to the initial state, MAN_{10n} , at which it can accept a new message (then goes to state MAN_{12}), forward the existing one or start migration (*move*). The last event requires

disconnection from the old node and performance of several internal actions.

At the state MAN_{12} the system includes two messages. It can generate a new message, start migration (by forwarding two existing messages) or forward the first message received. At the state MAN_{12} , the system is full, includes three messages and can only forward the message.

A called part of MAN system behaves very similarly. It also corresponds with buffer which includes maximum messages (three one-side buffers). The difference is only in migration request which is initiated by MM agent. At the initial state (MAN_{00}) the system is empty. Then, MM agent can start either forwarding the received messages (out action) or migration of the called agent. If the system is full, MAN_{03} , it includes three messages and can only receive a message. When a called agent migration starts, all messages in the system must be delivered to it on port *receive* before a new message is sent to port *out*.

$$\begin{aligned} MAN_{00} &\stackrel{def}{=} \text{out } v. MAN_{01}(v) + \tau. MAN_{00} \\ MAN_{01}(v_1) &\stackrel{def}{=} \text{out } v. MAN_{02}(v_1, v) + \text{receive}(v_1). \\ &MAN_{00} + \tau. \text{receive}(v_1). MAN_{00} \\ MAN_{02}(v_1, v_2) &\stackrel{def}{=} \text{out } v. MAN_{03}(v_1, v_2, v) + \text{receive}(v_1). \\ &MAN_{01}(v_2) + \tau. \text{receive}(v_1). \text{receive}(v_2). MAN_{00} \\ MAN_{03}(v_1, v_2, v_3) &\stackrel{def}{=} \text{receive}(v_1). MAN_{02}(v_2, v_3). \end{aligned}$$

Verification of the system is done by Mobility Workbench tool. Model checking of the system specification (System) with desired behavior of the system (MAN) is done applying weak open bisimulation with distinctions command (weqd). After starting the command, bisimulation found by MWB has 728 tuples for the incoming part and 323 tuples for the outgoing part of communication.

6. Conclusion

Formal specification and verification of a mobile agent migration and communication in a mobile agent network are reported. The specification is written in the π -calculus process algebra and verification is done by using model checking feature of the Mobility Workbench. The model consists of a multi-agent system whose agents communicate in a network allowing agent mobility. Specification is focused on agents communication while they migrate through the network. Special attention is paid to specifying mobility management agent responsible for agent addressing and message handling. The system is verified by the automated analysis of Mobility Workbench tool.

The system design has constant interaction between modeling and verification phases. Therefore, in reality, the border between these two phases does not exist.

Further studies will include modeling of more complex systems with more mobile agents and nodes.

References:

- [1] Lovrek, I., Sinkovic, V., Knowledge-Based Mobility Management in All-Mobile Network, *Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence*, LNAI 2774, Springer-Verlag, 2003, 661-667.
- [2] Lovrek, I., Sinkovic, V., Jezic, G., Communicating Agents in Mobile Agent Network, *Frontiers in Artificial Intelligence and Applications*, 82 (2002), 1; 126-130.
- [3] Kusek, M., G. Jezic, I. Ljubi, K. Mlinaric, I. Lovrek, S. Desic, O. Labor, A. Caric, D. Huljenic, "Mobile Agent Based Software Operation and Maintenance", *Proceedings of the 7th International Conference on Telecommunications ConTEL 2003*, pp. 601-608, Zagreb, 2003.
- [4] Holzmann, G. J., Design and Validation of Computer Protocols, *Prentice Hall*, 1991.
- [5] Bhat, G., Cleaveland, R., Verifying the ATM UNI 3.1 Signalling Protocol, *Personal communication*, 1997.
- [6] Jezic, G., I. Lovrek, G. Bhat, Verifying Multiparty Call in ATM UNI Signalling Protocol, *Recent Advances in Signal Processing and Communication*, Mastorakis, Nikos E. (Ed.), World Scientific and Engineering Society Press, 1999. pp. 324-329.
- [7] Zilio, S. D., Mobile Processes: a Commented Bibliography, *Proc. Modelling and Verification of Parallel Processes*, F. Cassez, C. Jard, B. Rozoy, M. Ryan (Eds.), Lecture Notes in Computer Science, Vol. 2067, Springer-Verlag., 2001, pp. 207-223.
- [8] Milner, R., Communicating and Mobile systems: the π -calculus, *Cambridge University Press*, 1999.
- [9] Sangiorgi, D., Walker, D., "The π -calculus: A Theory of Mobile Processes", *Cambridge University Press*, 2001.
- [10] Jezic, G., Communication and Collaboration in Mobile Agent Network, *Ph.D. Dissertation*, University of Zagreb, 2003. (in Croatian).
- [11] Victor, B., Moller, F., The Mobility Workbench – a Tool for the π -calculus, *Proc. of CAV*, 1994, pp. 428-440.
- [12] FIPA Agent Management Specification, *Foundation for Intelligent Physical Agents*, 2002, <http://www.fipa.org/specs/fipa00023>.
- [13] Cao, J., Feng, X., Lu, J., Das, S. K., Mailbox-Based Scheme for Mobile Agent Communications, *IEEE Computer*, 2002, pp. 54-60.
- [14] Orava, F., Parrow, J., An Algebraic Verification of a Mobile Network, *Formal Aspects of Computing*, 4(6), 1996, pp. 497-563.
- [15] Victor, B., A Verification Tool for the Polyadic π -Calculus, *Report DoCS 94/50* Department of Computer Systems, Uppsala University, Sweden, 1994.