

Parallelism in Mobile Agent Network

Vjekoslav Sinkovic
University of Zagreb

FER, Unska 3

HR-10000 Zagreb, Croatia

e-mail: vjekoslav.sinkovic@fer.hr

Ignac Lovrek

University of Zagreb

FER, Unska 3

HR-10000 Zagreb, Croatia

e-mail: ignac.lovrek@fer.hr

Mario Kusek

University of Zagreb

FER, Unska 3

HR-10000 Zagreb, Croatia

e-mail: mario.kusek@fer.hr

Abstract

The paper deals with parallel mobile agents and related performance evaluation framework. A model called mobile agent network is proposed. It includes a multi-agent system consisting of co-operating and communicating mobile agents, a set of processing nodes in which the agents perform services and a network that connects processing nodes and allows agent mobility. Parallelism in mobile agent network is based on parallel agents allocated to a single task requested from the environment. Mobile agent network is defined as queuing system where the agents represent information units to be served. Simulation based method for performance evaluation is included.

1. Introduction

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 [1, 2]. Use of the agents and agent's mobility in operation and management of networks, systems and services offers important advantages because of the network load reduction, increased asynchrony between the communicating entities and higher concurrency. Global end-to-end interactions are replaced by local interactions on a node visited by a mobile agent, the need for long reliable connections is reduced, bandwidth requirements are lower and repeated interactions less frequent.

This paper elaborates parallel agents in a mobile agent network [3, 4] and related performance evaluation framework. Mobile agent network consists of a multi-agent system and its agents co-operate, communicate and travel in a network. The users interact with the mobile agent network by requesting services. Each user request activates one or more parallel agents with a set of elementary services that correspond to users request. Agents can migrate autonomously from node to node, to perform some processing on behalf of a user. In order to introduce performance measures, a mobile agent network is described as a queuing system where an agent represents information unit to be served. Network flows of agents as well as the

parameters defining agent hosting, execution and migration are introduced.

The paper is organised as follows: parallel processing with mobile agents is presented in Section 2. Mobile agent network with parallel agents is defined in Section 3 and parallel agents elaborated in Section 4. Performance evaluation model is presented in Section 5, while Section 6 concludes the paper.

2. Parallel Processing with Mobile Agents

Related work includes different aspects of parallel processing with mobile agents. Agent behaviour defined by a set of concurrent tasks is given in [5]. A task is specified by finite automaton; i.e. the task must receive an event to make transition from one state to another. In transition or in state task can generate events. There are two types of events: internal and external. Internal events are sent to other tasks inside the same agent, while external events are sent to another agent. Besides communication with other agents, a task can interact with the environment by reading percepts or performing operations that affect the environment. In this model parallel processing is done by decomposing initial job, that agent must execute, into concurrent tasks.

In Master/Slave model [6] initial job is submitted to master agent, that creates a set of slave agents (also called worker agent) and sends them to different nodes in a network. There are two ways of operation: synchronous and asynchronous. In synchronous model, after sending slave agents, master agent waits for the atomic execution of all slave agents. In asynchronous model, master agent waits for one slave agent to return with partial results and makes decision about other agents, i.e. if master agent concludes that received results are satisfactory, it will terminate the rest of agents, and give results to the user. This model includes mechanisms for inter agent communication, event listeners, collective operations and load balancing.

Similar approach is used in [7]. The main difference is that slave agents can have more than one task to execute. Tasks are defined with task library and agents cannot execute anything outside the task library.

A team of agents is defined in [8]. Agent team consists of specialised agents, each performing particular role.

Agents co-operate as a team in order to achieve goal. Some basic agents are the following: hive agent, queen agent and scout agent. Hive agent is a stationary agent that receives user request and decomposes request into smaller tasks, which it gives to queen agent. Queen agent migrates to some node in the network, analyses local environment and starts task execution. The hive agent performs task apportionment and it needs to know where it can send queen agents. Scout agent is responsible for resource discovery; the hive agent periodically sends scout agent around the network.

3. Mobile Agent Network with Parallel Agents

A model called mobile agent network is proposed. Mobile agent network is represented by the following triple:

$$\{A, S, N\}$$

where

A - a multi-agent system consisting of co-operating and communicating mobile agents,

S - a set of processing nodes in which the agents perform services,

N - a network that connects processing nodes and allows agent mobility.

A mobile agent represents a user in a network. It can migrate autonomously from node to node, to perform some processing on behalf of a user. Structure of an agent is shown in Figure 1. It includes the following fields:

- owner,
- node,
- program part, and
- data part.

Agent's owner is defined by its identification and service request. Node data consists of actual node address and execution thread. Execution thread (the state of service execution) shows, for each elementary service, a node where it is executed. An elementary service is defined by

its program (procedure) and data (results).

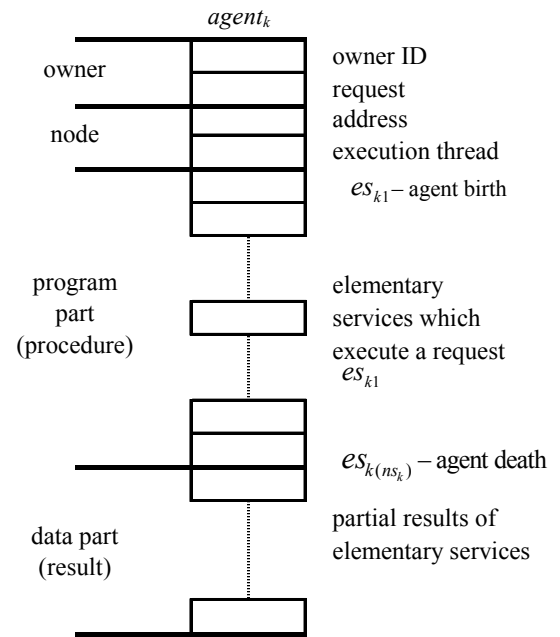


Figure 1. Agent Structure

A service, $service_k$, provided by $agent_k$ is composed of ns_k elementary services, es_{kl} , as follows:

$$ES_k = \{es_{k1}, es_{k2}, \dots, es_{kl}, \dots, es_{k(ns_k)}\}$$

Elementary services are similar to elementary tasks used in distributed parallel processing in telecommunications [9-12].

Each node, S_i , is characterised by a set of m_i service types, s_i , it supports.

$$s(S_i) = \{s_1, s_2, \dots, s_t, \dots, s_{nt}\}$$

At each host agent can execute services that satisfies $es_{kl} \in s(S_i)$.

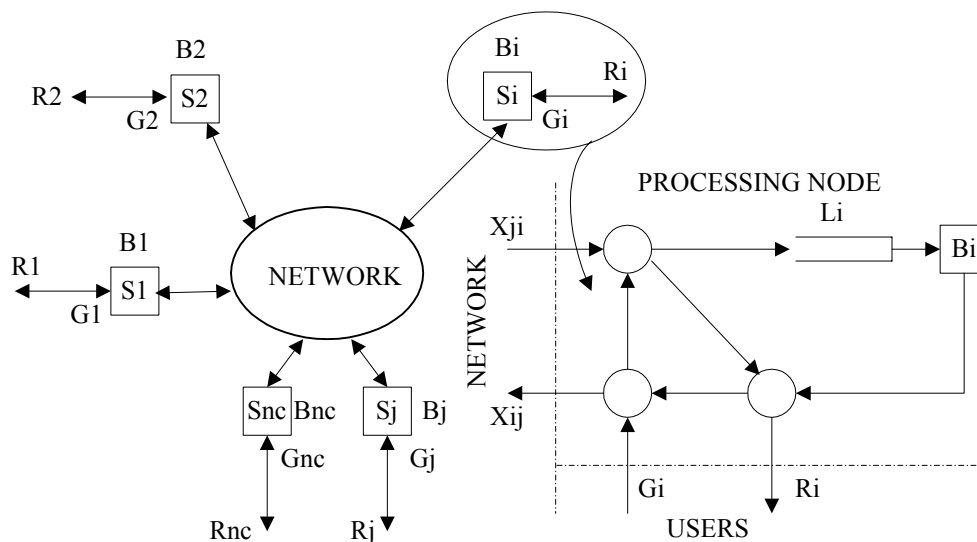


Figure 2. Mobile Agent Network Flows

The users interact with the mobile agent network by requesting the service (Figure 2). A node, S_i , collects the requests from its users and activates the parallel agents with elementary services that correspond to users request. Having completed all elementary services, the agents return the response to the user. Requests and responses form an input flow, G_i , and output flow, R_i respectively. Agent migration is defined by the flows between the network nodes, X_{ij} and X_{ji} . The agents accepted for execution at S_i node represent its internal flow, L_i . B_i denotes processing intensity of S_i node.

When an agent, $agent_k$, is created on node S_i , that node is its originating node. The first elementary service of each agent, es_{k1} , is "agent birth" and the last one, $es_{k(ns_k)}$, is "agent death". They must be executed at originating node. After "agent birth", agent starts to execute other elementary services. Having completed an elementary service, es_{kt} , $agent_k$ continues on the same node or moves to another one in order to execute next elementary service, $es_{k(l+1)}$.

4. Parallel Agents

Proposed model of parallelism in mobile agent network is based on parallel agents allocated to a single task (service) requested from the environment (users). Such set of agents originates at the same network node and returning to it after completing migration. Each agent can have one or more mutually independent subtasks (elementary services). When hosted by the same network node, agents can exchange results and reduce the number of non-executed elementary services. Examples that follow are used for explaining migration and execution of parallel agents.

Let us suppose four processing nodes $S_1 - S_4$ and a single agent $agent_1$ with 32 elementary tasks which can be executed on any node. Part of its migration and execution matrix is shown in Figure 3. Elements of migration and execution matrix are vectors:

$$\mathcal{E}_{1i} = \begin{pmatrix} i \\ t \end{pmatrix},$$

representing execution of es_{1i} where i defines the node label (i for S_i) where elementary service es_{1i} is handled and t gives the service type label of this elementary service (t for S_t).

For example, in 6th column is es_{16} of the type s_8 , which is executed at node S_7 .

$$\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 5 \end{pmatrix} \begin{pmatrix} 1 \\ 11 \end{pmatrix} \begin{pmatrix} 1 \\ 7 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 8 \end{pmatrix} \right) \cdot \cdot \cdot \left(\begin{pmatrix} 3 \\ 11 \end{pmatrix} \begin{pmatrix} 3 \\ 6 \end{pmatrix} \begin{pmatrix} 3 \\ 5 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 \\ 10 \end{pmatrix} \begin{pmatrix} 1 \\ 12 \end{pmatrix} \right)$$

Figure 3. Migration and Execution Matrix for Single Agent

It is assumed that execution time for each elementary service and agent transfer time for all pairs of nodes are the same, and equal to Δt . So the response time equals $38\Delta t$ ($32\Delta t$ for execution of elementary services and $6\Delta t$ for transfer between processing nodes).

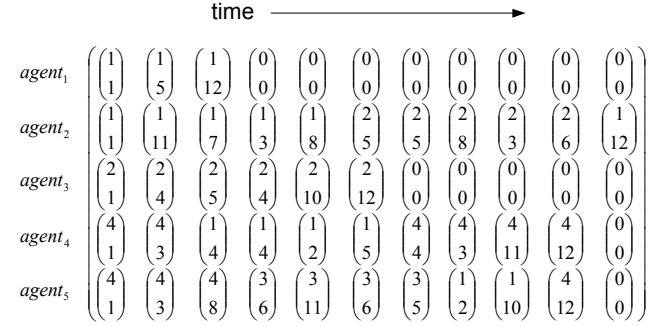


Figure 4. Migration and Execution Matrix for Parallel Agents

Let us suppose further on, five parallel agents $agent_1 - agent_5$ serving the same request. Because the first and the last elementary service are always the same ("agent birth" and "agent death"), 30 elementary tasks from the Figure 3 remain for allocation to the agents. An example is shown in Figure 4. A row k corresponds to $agent_k$ and describes its migration and execution path. The first and the last elementary service in each agent are "agent birth" and "agent death". For instance, $agent_4$ with 10 elementary services starts at S_4 node. The first two elementary services (es_{41} , es_{42}) are served at this node and the agent is transferred to S_1 node where next four elementary services ($es_{43}-es_{46}$) are completed. The last four elementary services ($es_{47}-es_{410}$) are accomplished at S_4 node.

Time packed execution path is shown in Figure 5, where each processing node has 5 separate vectors. Vector τ_{ki} represents execution of $agent_k$ on node S_i , as follows:

$$\tau_{ki} = \begin{pmatrix} \tau_{ki1} \\ \vdots \\ \tau_{kip} \\ \vdots \\ \tau_{kiP} \end{pmatrix},$$

where τ_{kip} defines type of an elementary service es_{ki} executed at S_i in time interval p . A label 0 represents an empty interval. Rows corresponds to time intervals $1 - P$.

Time to execute $agent_1$ with 3 elementary services is $5\Delta t$ (Figure 5). All elementary tasks are performed on node S_1 . The last elementary service of $agent_1$ is placed in the first vector in 5th row. Average execution time of all agents is $12.2\Delta t$. Node S_1 has executed 15 elementary services and its load is 83.3%. Node load is defined as the percentage of executed elementary services on the node, divided by the

number of time intervals P . Average node load in the system is 55.6%.

The above 5 agents are mobile agent team that execute 30 elementary services and for that they need $18\Delta t$ time, which is 47.4% of time for the same set of elementary services executed with one agent.

Total number of executed elementary services in system is 40 because of “agent birth” and “agent death” elementary services.

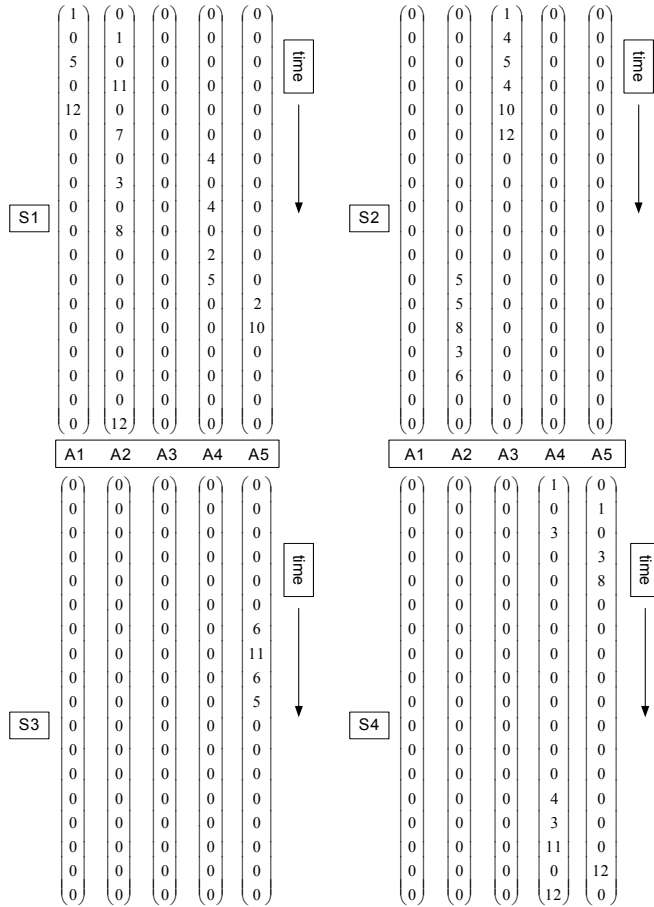


Figure 5. Time Packed Execution Path

When more than one agents are on the same node and one agent need to execute elementary service that another agent has already completed and has result, the first agent can undertake result and save execution time. In Figure 6 is example for that.

For example at node S_4 $agent_5$ is first executing elementary service of the service type s_1 and after that of s_8 . It is shown in the migration and execution matrix (Figure 4) that after elementary service of the type s_1 $agent_5$ must execute elementary service of the type s_3 . Because $agent_4$ is on the same node as $agent_5$, and $agent_4$ has executed elementary service of the type s_3 then $agent_5$ has undertaken the result of the service and executed next elementary service es_8 from migration and execution matrix.

Average time for all agents in optimised execution path is $9.6\Delta t$ and average node load is 53.8%.

Total number of executed elementary services in system is 28. All this figures show the benefit of such optimisation.

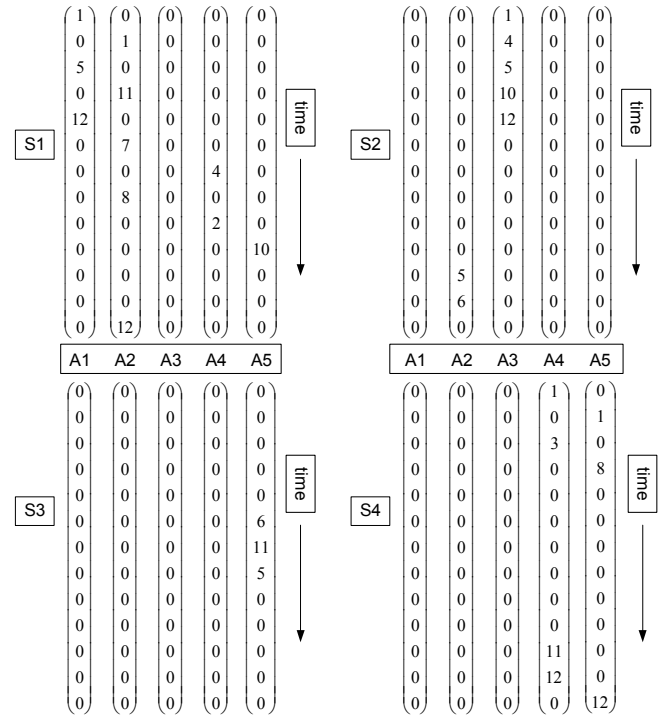


Figure 6. Optimised Time Packed Execution Path

5. Performance Evaluation

In order to introduce performance evaluation capabilities, mobile agent network is described as a queuing system where the agents represent information units to be served (Figure 7).

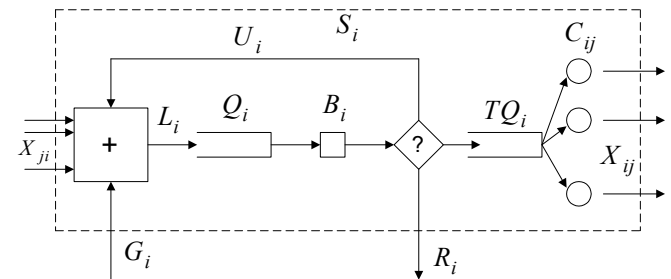


Figure 7. Network Node Structure

The nodes represent the servers capable of hosting and executing the agents. The node processing capacity is B_i . The node S_i receives network flows X_{ji} and input flow G_i . Those flows are summarised as L_i and sent to queue Q_i . When an $agent_k$ from queue comes in order, then processor executes elementary service es_{kl} from $agent_k$. After processing, an agent decides to execute next elementary

service $es_{k(l+1)}$ on the same node (U_i flow) or migrate to another node and executes next elementary service there. Migrating agent is put in migrating queue TQ_i . When the $agent_k$ comes in order to be migrated then it is serialised and transferred to node S_j . Transmission capacity between nodes S_i and S_j is C_{ij} . If $agent_k$ has computed the last elementary service es_{kl} then it is put in output R_i flow.

The actions performed during agent migration and execution are defined by the parameters shown in Figure 8. Agent migration from S_i to S_j , is described by agent transfer time:

$$T_{ij} = t_{pi} + t_{ij} + t_{aj}$$

where

t_{pi} - agent preparation time needed for agent serialization and other operations related to agent migration,

t_{ij} - agent communication time needed for agent transfer from S_i to S_j ,

t_{aj} - agent activation time that includes agent reception, de-serialisation and restart.

Agent handling at the elementary service level at S_j is described by holding time:

$$t_{qj} = t_{cj} + t_{wj} + t_{sj},$$

where

t_{cj} - interagent communication time, i.e. the time an agent spends in the S_j searching for result of elementary service performed by another agent,

t_{wj} - waiting time, i.e. the time an agent spends in the S_j queue expecting elementary service execution,

t_{sj} - serving time, i.e. the time needed for elementary service execution at S_j .

All elementary services are the same with respect to serving time, which equals Δt . All parameters are expressed in discrete time units, Δt .

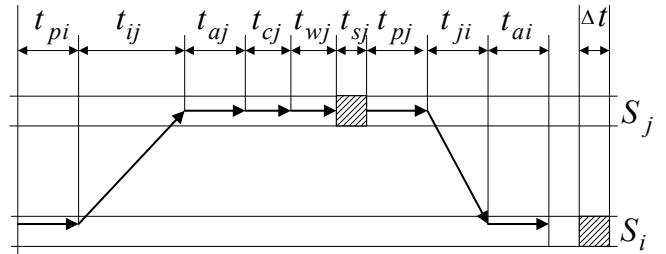


Figure 8. Agent Transfer and Holding Time

Simulation based method for performance evaluation is proposed. Mobile agents network features are described as follows. User requests are random events as well as activation of the parallel agents capable of handling the services. Agent execution proceeded by its migration from its originating to destination node, and response to a user, are also stochastic processes. Agent migration feature is defined as follows:

- An agent $agent_k$ handles ns_k elementary services es_{kl} , where ns_k represents random variable bounded by NS ,
- After completing the elementary service, es_{kl} , at S_i node, the node S_j for hosting $agent_k$ and serving next elementary service, $es_{k(l+1)}$, is selected randomly.

Figure 9 shows simulation results for the mobile agent

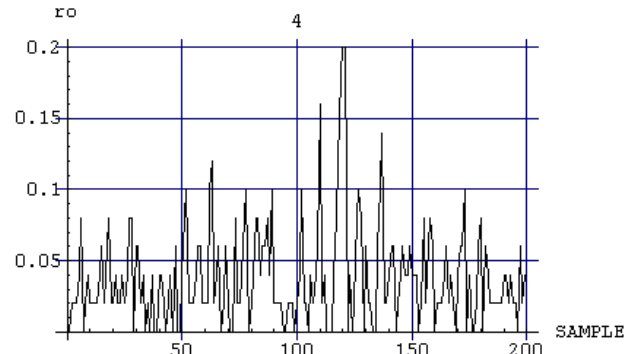
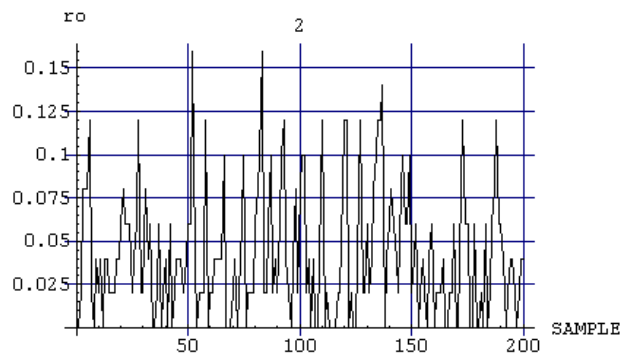
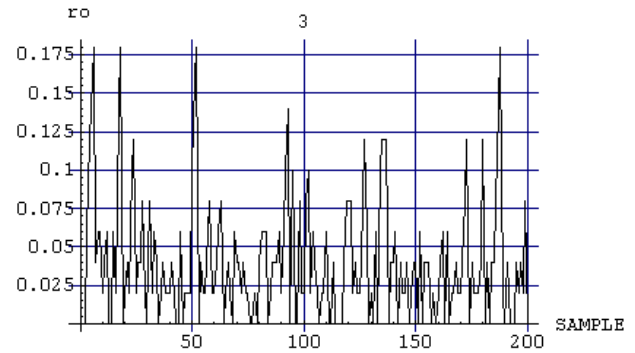
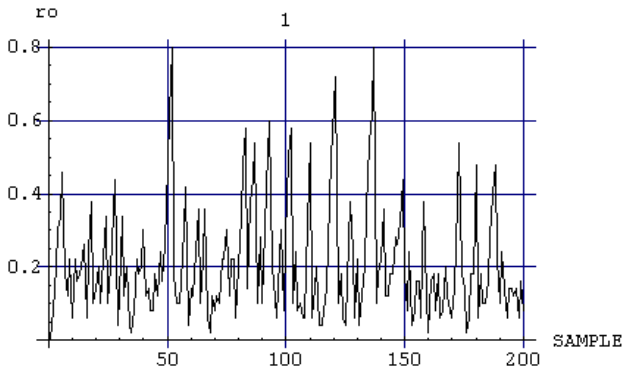


Figure 9. Processor Load at Sites S_1 , S_2 , S_3 and S_4

network with 4 nodes. Parameter ro represents node load. Simulation was performed over 200 samples of agent bursts (parallel agents) that represent user requests. Burst interarrival time follows exponential distribution. Agent arrival intensity and mean burst interarrival time define the input agent flow generation. The number of the agents in a burst is a random value, as well as the number of elementary services carried by an agent. Uniform distribution is applied to both parameters. Each burst is quantified by two parameters: number of agents and total number of elementary services.

Conclusion

Model of mobile agent network with parallel agent is introduced. Parallel operations within multi-agent system are specified. Further on, a mobile agent network is described as a queuing system where an agent represents an information unit to be served. Agent hosting, execution and migration are defined as stochastic processes. Simulation-based method for evaluation of performance is proposed.

Future work will include improvements of formal model and performance evaluation of parallel agent execution. Also problems of dispatching tasks to mobile agent teams will be investigated.

References

- [1] W. Cockarine, M. Zyda, *Mobile Agents*, Manning, Greenwich, 1998.
- [2] W. Brenner, R. Zarnikow and H. Wittig, *Intelligent Software Agents - Foundations and Applications*, Springer, Berlin, 1998.
- [3] V. Sinkovic and I. Lovrek, Generic Model of a Mobile Agent Network Suitable for Performance Evaluation, *Proceedings KES'2000 Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, Brighton, UK, 2000, pp. 675-678.
- [4] I. Lovrek and V. Sinković, Performance Evaluation of Mobile Agent Network, *Proceedings KES'2001 Fifth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, Osaka, Japan, 2001, pp. 924-928.
- [5] Scott A. DeLoach, Specifying agent behaviour as concurrent tasks, *Proceedings of the fifth International Conference on Autonomous Agents*, Montreal, Canada, 2001, pp. 102-103.
- [6] L. M. Silva, V. Batista, P. Martins, and G. Soares, Using Mobile Agents for Parallel Processing, *Proceedings of the International Symposium on Distributed Objects and Applications*, Edinburgh, United Kingdom, 1999.
- [7] C. Panayiotou, G. Samaras, E. Pitoura, and P. Evripidou, Parallel Computing Using Java Mobile Agents, *Proceedings of the 25th Euromicro Conference (EUROMICRO '99)*, Milan, Italy, 1999.
- [8] R. Ghanea-Hercock, J. C. Collis, and D. T. Ndumu, Co-operating Mobile Agents for Distributed Parallel Processing, *Proceedings of the Third Annual Conference on Autonomous Agents*, ACM Press, New York, USA, 1999, pp. 398-399.
- [9] Sinkovic V. and Lovrek. I., An Approach to Massively Parallel Call and Service Processing in Telecommunications, *Proceedings First International Conference on Massively Parallel Computer Systems MPCS 94*, Ischia, 1994, pp. 533-537.
- [10] Lovrek. I. and Sinkovic V., Load Balancing Potential of Distributed Parallel Call and Service Processing in Telecommunications, *Proceedings Second International Conference on Massively Parallel Computer Systems MPCS 96*, Ischia, 1996, pp. 366-373.
- [11] G. Nemeth, I. Lovrek and V. Sinkovic, Scheduling Problems in Parallel Systems for Telecommunications, *Computing*, 58(1997), pp. 199-223.
- [12] Sinkovic V., Lovrek. I. and Desic S., Processing Speedup and Load Balancing in Distributed Parallel Call and Service Control in Telecommunications, *Proceedings Third International Conference on Massively Parallel Computer Systems MPCS 98*, Colorado Springs, 1998.