Automation, Control and Intelligent Systems
2014; 2(6): 105-111
Published online December 05, 2014 (http://www.sciencepublishinggroup.com/j/acis)
doi: 10.11648/j.acis.20140206.12
ISSN: 2328-5583 (Print); ISSN: 2328-5591 (Online)

Performance models preventing multi-agent systems from overloading computational resources

Petr Kadera[1, *], Petr Novak[1, 2], Vaclav Jirkovsky[1, 3], Pavel Vrba[1]

[1]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, CZ-169 00, Prague, Czech Republic
[2]Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems, Vienna University of Technology, A-1040, Vienna, Austria
[3]Rockwell Automation Research and Development Center, CZ-150 00, Prague, Czech Republic

Email address:
petr.kadera@fel.cvut.cz (P. Kadera), novakp46@fel.cvut.cz (P. Novak), jirkova@fel.cvut.cz (V. Jirkovsky), pavel.vrba@ciirc.cvut.cz (P. Vrba)

To cite this article:
Petr Kadera, Petr Novak, Vaclav Jirkovsky, Pavel Vrba. Performance Models Preventing Multi-Agent Systems from Overloading Computational Resources. *Automation, Control and Intelligent Systems*. Vol. 2, No. 6, 2014, pp. 105-111. doi: 10.11648/j.acis.20140206.12

Abstract: Multi-Agent Systems (MASs) suffer from low immunity against burst of arrival requests which can result in a permanent outage of such systems. This factor limits the suitability of MASs for control of real-world manufacturing systems with strict requirements on performance and reliability. This manuscript explains the origins of the performance degradation of MASs based on Contract-Net Protocol and proposes a method that protects the systems against the destructive effect of temporal overloads. The proposed method continuously observes the communication among agents and analyzes it in order to identify possible saturation of a system resource. If triggering a new action saturates a system resource, the carrying out of the action will be postponed. The impacts of the method are demonstrated on a test-bed consisted of six mini-computers Raspberry Pi. It shows that the proposed method avoids overloading of the system and thus guarantees a specific system throughput effectively and efficiently.

Keywords: Holonic Systems, Multi-Agent Systems, Robustness, Reconfigurable Systems, Software Agents

1. Introduction

Multi-Agents Systems (MASs) have provided a new abstraction paradigm for designing distributed and flexible industrial control systems emphasizing attributes such as autonomy, robustness, survivability, adaptation, and reconfiguration [1][1][3]. Agents find optimal solutions at runtime, which eliminates the need for preparing control strategies for all possible scenarios in advance.

A shift of MASs from laboratory research to real world deployment is slower than expected. This is caused by multiple factors; one of them is a group of performance-related problems of MASs caused by overload of computational resources. This causes hardly predictable delays or even prevents the MASs from converge to a solution due to expired communication timeouts. This manuscript proposes a method that protects the MASs based on Contract-Net Protocol (CNP) against overloading caused by bursts of requests. Providing such a protection is of the highest importance because even a temporal burst of requests can transfer the MAS to such a state from which the system is not able to recover. Many concurrent actions overload resources and prolong the execution times. This may lead to the expiration of the negotiation timeouts. Then, agents usually invoke new attempts to cooperate, but it repetitively ends up with not passing the timeouts due to the reoccurring overloads. This forms a never ending loop, from which the agents cannot escape. In practice, this situation frequently occurs in the system setup mode, when the initial state has to be reflected in agent actions, as well as in the operation mode, when external events come with excessive frequency.

In general, these problems are caused by saturation of computational resources. Overloaded resources decrease the agent responsiveness which might end up with exceeding the communication timeouts. This fact closely corresponds with another problem of MASs: it is difficult to find the optimal setup of communication timeouts, which the agents use to bound their waiting for responses. Too long timeouts decrease the performance by waiting for messages that never come (e.g. from a broken agent). On the other hand, the communication timeouts cannot be too short either, because it might disallow the system to converge to the best solution as have been many times experienced during the work on the Chilled Water System (CWS) application [4]. Moreover, the optimal setup is specific for each system configuration (including number of agents, computational performance of used hardware, and system load). Thus, any hardware or software change of the well-tuned system has to be followed by a new timeout setup.

This manuscript proposes a method which replaces the fine tuning of communication timeouts by a congestion control mechanism that prevents the system from entering overloaded operational regimes. The method breaks the relation between timeout settings and the current system load. This is achieved by the observation of the agent communication and its analysis which provides information how each initial request loads each of the system resources (e.g. a CPU). These parameters form a Loading matrix, which is used as input for the Operational analysis. It is an analytical method that identifies among all resources potential system bottlenecks. The saturation load in terms of the frequency of arrival requests is then identified for the bottleneck candidates.

The low computational complexity makes this method applicable at runtime, where it extends capabilities of the regular Directory Facilitator (DF). Consequently, the extended DF can spread the possible burst of request into a longer time period, in order to prevent any part of the system against saturation. The method is directly applicable to MASs utilizing the Contract-Net Protocol (CNP) [5] or its extension Plan Commit Execute (PCE) protocol [6], because the cooperation in such systems forms chains, where initiators can be identified and then the successors can be traced in order to estimate the overall impact of the initial request. The method is further limited only to such systems that use DFs for starting cooperation between agents.

Utilization of the method is demonstrated on a segment of CWS containing one service agent (requests cold water), four valve agents (connect water piping sections), and one chiller agent (provides cold water). The experimental evaluation was done on the test-bed consisted of six minicomputers Raspberry Pi Model B (see Figure 1), which host Jade agents.

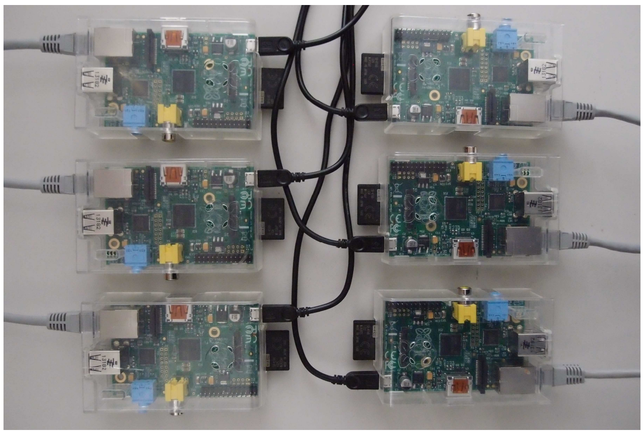
Figure 1. *Testbed containing six minicomputers Raspberry Pi Model B.*

2. Related Work

This section introduces selected methods, approaches, and tools that increased the acceptance of MASs and distributed embedded systems by industrial enterprises.

2.1. Methodologies for MASs

A lot of attention in the area of the development of a methodology for holonic systems was paid within the research program for Intelligent Manufacturing Systems (IMS). Within its activities, two holonic architectures were developed – PROSA [7] and ADACOR [8]. Both introduce sets of guidelines to decompose manufacturing control functions into communities of autonomous and cooperative entities called holons. Recently, methods for validation of agent-based manufacturing systems have become investigated. ANEMONA [9] presents means for functional validation of multi-agent architectures in conformance with specific trade requirements.

2.2. Model-Based Diagnostics

MABLE [10] is a conventional imperative programming language which is extended by constructs from MASs. The agents designed in MABLE maintain their social knowledge using linear temporal belief-desire-intention logic. The major advantage of this approach is the ability to formally prove that any interaction of agents will not lead to a fault state. On the other hand, the fundamental disadvantage and perhaps the stopper for the wider spread of this technique is the limited set of constructs that an agent can use. In other words, the agent has to be designed in the way suitable for MABEL from the very beginning.

2.3. Formal Time Analysis for Embedded System

The domain of embedded systems is facing a dramatic increase of the network complexity. For example, modern automotive control systems contain more than fifty electronic control units (ECUs) that are produced by various suppliers [11]. The units are inter-connected via a communication network representing a shared resource. It is necessary to assure that a potential conflict in usage of the shared resource would not lead to a dangerous situation. It means, for example, the function of the Anti-lock braking system (ABS) in a car must not be harmed by increased communication of other systems. Similar problem and requirements come from the aircraft industry and also from the designers of multi-processor systems [12]. In general, networked or distributed systems can be characterized by observing a high amount of data flows within the network. To address the challenges posed by the increased network complexity the Network Calculus [13] and its extension Realtime calculus [14] were introduced. Network Calculus enables evaluation of timing properties of data flows in communication networks. Realtime Calculus extends this concept to make it suitable for real-time embedded systems. The basic idea behind these two approaches is to substitute individual events by data flows called event streams. The validation problem is then transformed into the examination of flows, which can be solved by efficient methods.

2.4. Qualitative and Quantitative Analysis of Industrial Multi-Agent Systems

The investigation of methods for validation of MASs was addressed by the EU FP7 research project GRACE [15]. One of the project outputs is the methodology for qualitative analysis based on Petri net modeling notation. The behavior of each agent is represented as a single Petri net which can be verified by the regular methods to find out whether the model is bounded (the resource can only execute one operation at time), reversible (the agent can reinitiate by itself) or out of deadlocks (the agent can make at any state an action). The extension of these models with concept of time provides methods for quantitative analysis of multi-agent systems. The transitions are extended with the time parameters to capture the times of transition activations. Such a simulation shows the evolution of the tokens over places and over the time. The complete information about the progress of the agent behavior is summarized with a Gantt chart. Unfortunately, the development of the Petri net models is a time-consuming process, which requires specialized skills.

2.5. Supportive Tools for Development of Multi-Agent Systems

Debugging and tuning of MASs is a challenging process that cannot be tackled by methods used for monolithic applications such as debuggers and profilers. Particularly, the multi-agent applications cannot be debugged step by step due to the asynchronous communication between components. Instead, logging mechanisms are used to capture the time-lines of system events that are afterwards analyzed by the programmer. Technically, a log is captured by a specialized meta-agent that is usually called sniffer. A basic sniffer is part of the Jade platform. Advanced features to visualize the workflows provides Java Sniffer [16].

3. Scheduling of Operations in MASs

The prevention of MASs from entering overloaded operational regimes is based on observation of communication among agents. The fundamental part of the method is the automated identification of loading matrices from communication logs. In more details, the entire method consists of six steps. First, the communication produced by a MAS is logged and divided into groups according to the initial events, i.e., every group consists of messages that are successors of the same initial event. Second, the messages in each group are causally ordered (from initiators to participants) into tree structures. Such trees are called workflows and the initial events are their roots. Third, the workflows are analyzed to identify the loading matrix. Fourth, operational analysis is utilized in order to identify the bottleneck candidates. Fifth, the DF is extended by scheduling features which prevent the system from triggering too many activities in parallel. Sixth, the comparison of the original and the improved MASs are presented to illustrate the contribution of the method.

3.1. Communication Analysis

Many MAS platforms including JADE and Autonomous Cooperative System (ACS) support logging (sniffing) messages sent among agents. The proposed method builds on this feature and develops a new sniffer that observes the overall agent communication in a similar way as other sniffers, but it adds the time analyzes of the logged messages, whereas regular sniffers focus mainly on the visualization of the communication. The new sniffer analysis causal relations and timing of messages in order to obtain workflows, i.e., knowledge what are the initial messages and what are their successors.

The global behavior of MASs emerges from local interactions between components. These interactions have a causality defined by roles of communicating agents: initiator → participant. Because the participants can also become to be initiators to re-distribute the received requests, the interactions form cooperation chains called workflows. It is necessary to identify these workflows to understand the impact of the beginning request on the whole system. These initial events are called Primary actions because they are triggered by internal events in agents. They drive the agent's proactivity as they cause that the agents initiate new conversations as reactions to perceptions done inside of the agents. A typical example from the industrial domain is an agent that receives notifications from the low-level control about a change of a data tag value. It causes a reaction of the agent in terms of creating a new conversation with other agents. The Secondary actions are reactions on received messages coming from either Primary or another Secondary action.

Messages sent among agents are composed according to the FIPA ACL specification [17]. This specification defines a set of mandatory parameters that each message contains. These parameters are used to identify who the sender is and which conversation the message belongs to. Such an identification is necessary to enable agents to work on multiple tasks in parallel without mixing up messages coming from different conversations. To enable the backward communication traceability, agents use parameter "Reply With" to add additional pieces of information into messages. The content of this field is composed iteratively by all agents participating on the particular negotiation. The original seed of the parameter creates the initiator of the communication and stores in it its name and the number of the conversation, in which this agent takes part (e.g.: svc1@100:0). The next participant extends the content of this field by the "~" character and its own identifier (e.g. svc1@100:0~acp@100:310) and so on. This annotation is sufficient for the reconstruction of the workflows.

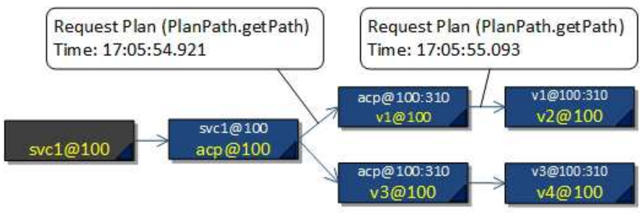

Figure 2. *Example of a workflow that illustrates a piece of negotiation in CWS application. The bubble labels highlight parts of the messages sent between agents.*

In general, a communication log consists of multiple workflows. The first step of the communication analysis is the organization of the messages into individual workflows. Using terminology of the graph theory, a workflow is a tree (see Figure 2) and the whole communication is a forest of such trees. The root of each workflow is the initiator of the particular communication. Intermediate nodes represent agents that are not capable to satisfy the received request on their own, but are able to inquire other agents. Finally, workflow leafs represent agents, who (i) can fully satisfy the request or (ii) cannot and even are not able to involve into the negotiation process any other participants.

The next step is the computation of the loading matrix. It is a table (see Figure 3) containing information how each initial request (Customer class in terminology of performance models) Cr loads a resource (Job Center in terminology of performance models) Ji. The load is derived from time differences between input and output messages.

$$\begin{array}{c||cccc}
 & C_1 & C_2 & \cdots & C_R \\
\hline
J_1 & L_{11} & L_{12} & \cdots & L_{1R} \\
J_2 & L_{21} & L_{22} & \cdots & L_{2R} \\
\vdots & & & \ddots & \\
J_M & L_{M1} & L_{M2} & \cdots & L_{MR}
\end{array}$$

Figure 3. *Loading Matrix.*

The computation of a cell of the Loading Matrix is illustrated on a concrete example. First, the communication is captured by Sniffer. Second, the messages are organized into workflows. Third, the load is computed from the time differences. For example, the cell in the matrix corresponding to the column for customer “Request from SVC1@100” and the row representing resource which hosts agent “v1@100” contains value 0.172 (17:05:55.093 - 17:05:54.921), i.e., time in seconds representing the time distance between the input and output message. The matrix is constructed row by row and each row represents a workflow. If the workflow for a specific initial request is received again, to old value are overwritten by the new ones. Whenever the Loading Matrix is changed, the updated values are sent to the DF.

3.2. The DF for Prevention against Overload

The proposed method benefits from a DF overview of ongoing activities, because the DF enabled to start all of them. This is combined with the loading matrix communicated from the sniffer and serves as input for the Operational analysis in details described in the next paragraphs.

The Operational analysis [18][19] based on operational laws is the most straightforward analytic technique for performance considerations. It outperforms other approaches in the speed, it is simple for implementation and does not require any specialized skills. On the other hand, this notation can be used only for systems with homogeneous workload, i.e. the behavior of system components (jobs, resources) is time invariant. Beside this, operational laws cannot describe any notion of synchronization or exclusive access.

Utilization law: The utilization is equal to the product of the throughput and the mean service time.

Because the method proposed in this paper uses only the utilization law, we will not introduce the other two operational laws (i.e., Little's law and Forced flow law) in detail.

The operational analysis was selected due to its low computational complexity. The operational laws, as they were introduced in [18], are directly applicable only in single-class case, but the multi-class cases require extension of the notation to handle various load coming from various customer classes. This work adopts and further extends method introduced by Casale [20].

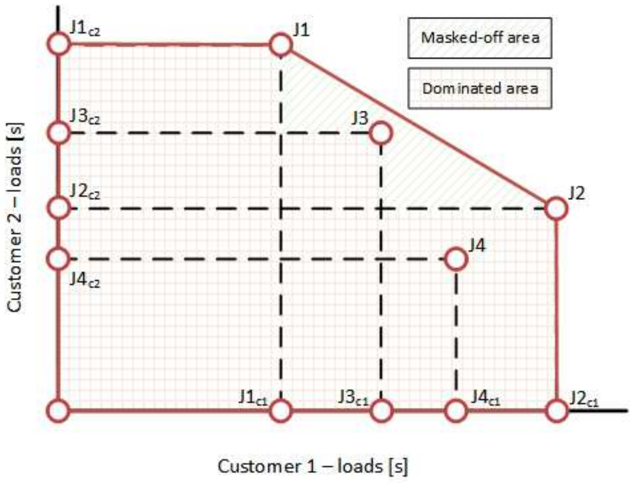

Figure 4. *Graphical representation of a loading matrix meaning.*

The key idea of the proposed method is that all system resources should be prevented from entering a saturated operational mode, which decreases the throughput of the particular resource and consequently of the whole system. To achieve this, the loading matrix, respectively its graphical representation, is used. Figure 4 depicts graphical representation of a loading matrix of a system with two customer classes. Points J1, J2, J3, and J4 represent individual job centers (in this case CPUs hosting agents). Their x- resp. y-coordinate represents the load imposed by customer of class 1 resp. 2. It is worth noticing, that the non-bottleneck JCs can be of two types. The first are dominated by another JC (e.g. J4 is dominated by J2) or they might be masked off by a combination of other centers (e.g. J3 is masked off by the combination of J1 and J2). The method constructs the convex hull around all job centers. The job centers placed on a facet of the hull are the first ones facing the danger of saturation, therefore it is sufficient to keep out of the saturation these ones to guarantee that no other JC becomes saturated. The formal definition of this statement and its proof can be found in [20].

The transformation to the λ-space, where λ_r denotes the arrival frequency of r-customer class, is needed for further analysis.

$\lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_r\}$ is the vector of arrival frequencies of all customer classes. The transformation is based on the following equation:

$$U_i(\lambda) = \lambda_r * L_{ir} \tag{1}$$

For non-saturated stations holds:

$$U_i(\lambda) \leq 1 \tag{2}$$

where i iterates over the set of all bottleneck candidates as were identified in the previous step.

The set of equations (2) defines the region, for which holds that none of the resources is saturated.

3.3. Scheduling Extension of the DF

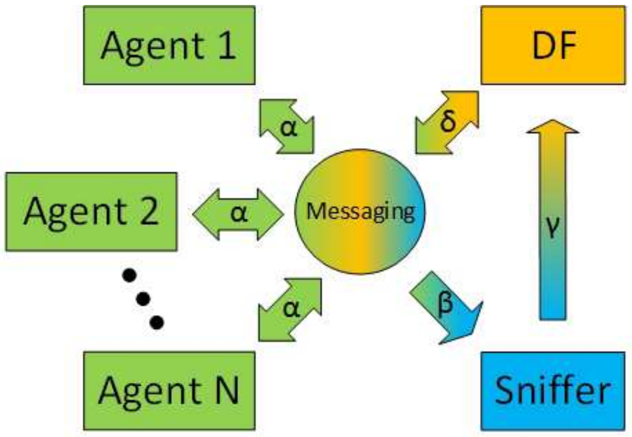

Figure 5. *Architecture of a MAS with Sniffer and Extended DF, which postpones the impracticable actions.*

Attention was paid to the interoperability of the designed method with existing systems. The final implementation comprehends extension of two meta-agents — Sniffer and DF (see Figure 5). The first one is an extension of the Java Sniffer, which logs the messages and analyzes them in order to derive the loading matrix *L*. The matrix is then communicated via the regular messaging channel to the DF. This meta-agent utilizes the matrix to detect, whether the series of requests arriving to the system causes saturation of any JC — in this case it is a computer hosting an agent. If the danger of saturation is detected, the request is postponed to match the maximum frequency under given conditions.

The Sniffer's part contains the regular and freely available Java Sniffer, which is responsible for capturing the messages, and further the newly developed extension, which computes the load matrix from the timestamps of corresponding messages. Beside this, the extension communicates its observations to the DF. All components of this part are written in pure Java and use only common libraries.

The DF's part extends the standard Jade's DF in three main parts. The first one is the register implemented as a HashMap, where keys are the request types and the values are timestamps denoting the time of the last occurrence of the request. A request type is a unique combination of an agent and the requested service. For instance, if agent "A1" requests service "cooling" then the request type is "$A1_{cooling}$." The second part is related to the computation of the convex hull. The computation itself is done in Matlab. The Matlab version R2014a provides function for computation of convex hull in n-dimensional space:

The input L is the loading matrix (the number of rows is the number of JCs, columns refer to the dimension of the space) and the output K is a matrix[x,y] - x is number convex hull facets, y is the dimension. In other words, the first row of matrix K contains indices of points from L that demarcate the first facet. The bottleneck candidates are such points that appears among the facet members. Set of constraints for these possible bottlenecks is created according to the set of equations (2).

When the DF is requested by an agent for a provider of a particular service, the current **λ** is computed as follows:

For the currently received request — of n-th customer class,

$$\lambda_i = 1/(t_{now} - t_{iL1}),\ i = n \tag{3}$$

$$\lambda_i = 1/(\max(t_{now} - t_{iL1},\ t_{iL1} - t_{iL2})),\ i \neq n \tag{4}$$

where t_{now} is the current time, t_{iL1} is the time, when the task i was triggered last time and t_{iL2} is the time, when the task i was triggered before the last occurrence. The procedure is illustrated in Figure 6.

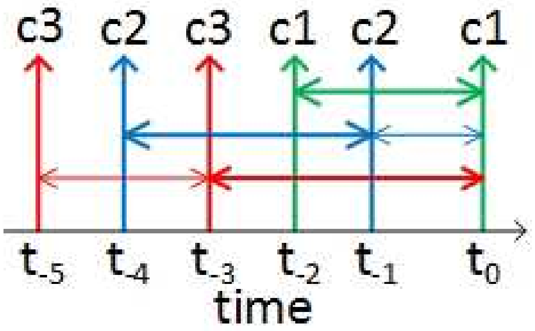

Figure 6. *Computation of arrival frequencies.*

Afterwards, the satisfaction of all constrains is tested. If none of them is violated, the request is handled immediately. Otherwise, the DF repeats the test after a relaxation time T until all constrains are satisfied.

4. Experimental Evaluation

The first experiment confirms that the behavior of the real distributed systems is coherent with the simulation results. Using the test bed, we set up system with six Jade agents representing a part of the CWS application. One agent represents a service (the agent asking for cooling water), one agent represents a chiller (the agent providing cooling water) and four valve agents (the agents interconnecting segments of the water-piping system). The cooperation schema is depicted in Figure 2. The initiator of the communication is the service agent *svc1*. It asks the DF for contacts to agents providing cooling water. The DF provides contact (i.e., an agent id used for addressing messages within the JADE platform) to the chiller *acp.* The chiller communicates with the valves (*v1*, *v2*, *v3*, and *v4*) in order to arrange the transport of the cooling water to the *svc1*.

The plot (see Figure 7) depicts the relation between frequency of request entering the system (Input Frequency) and the frequency of finishing the jobs - system throughput. The measured characteristic confirms the existence of the throughput maximum.

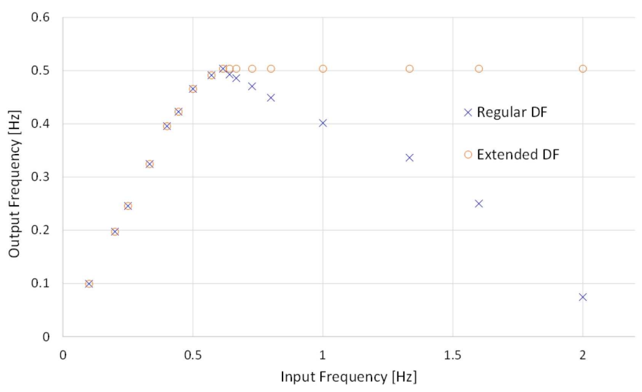

Figure 7. *Throughput with One Customer Class.*

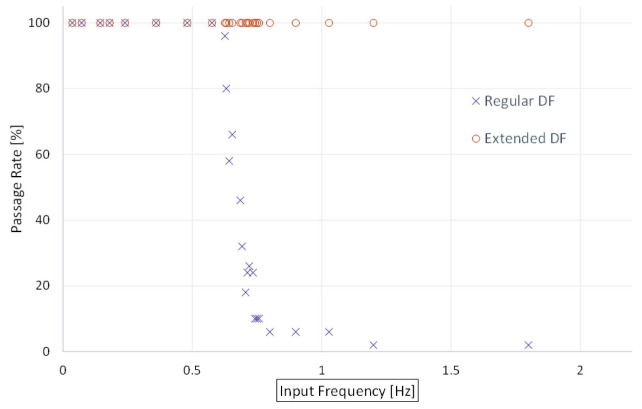

Figure 8. *Passage Rate*

The positive impact of using the extended version of the DF is clearly shown in Figure 8. The points marked by crosses represents measurements done with the regular JADE system and it demonstrates that after crossing a certain frequency the amount of jobs passed by a certain deadline steeply decreases. On the other hand, the modified version of the DF regulates the frequency of the input jobs, which leads the preserving the passing rate at 100 %.

The further experiments were focused on systems with more customer classes. We present the contribution of the proposed approach on a 2-customer system, i.e., on a system, where two types of customer requests occur. It was created by extending the previous 1-customer experiment by a new type of request, but still using the same test bed. The measurements go along with the previous results regarding the performance degradation, but on top of that also demonstrates the strong influence across input frequency of individual classes (see Figure 9 and Figure 10).

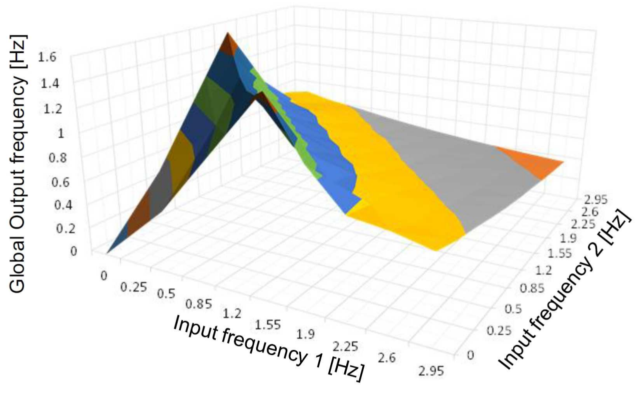

Figure 9. *Throughput with Two Customer Classes - Regular DF.*

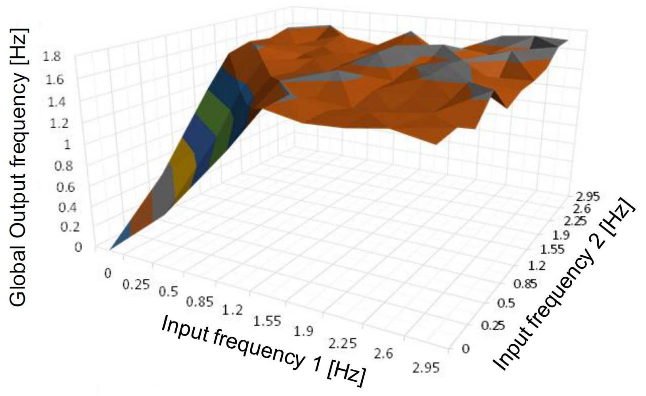

Figure 10. *Throughput with Two Customer Classes - Extended DF.*

5. Conclusion

This manuscript proposes a method to fill the gap in the design methodology for MASs related to the performance and responsiveness of the system. The method prevents a MAS from entering such operational regimes that would saturate a system resource. To achieve this, the communication between agents is observed (i) to obtain the loading matrix and (ii) to observe the frequency of tasks arriving to the MAS. Based on the analysis of the derived loading matrix, the method excludes from the further consideration such resources that cannot become bottlenecks. The loading parameters of the potential bottlenecks generate a set of constrains on frequencies of arriving tasks that cannot be exceeded unless some resources are saturated. The extended DF provides features to guarantee fulfillment of all constrains by delaying new tasks if needed. The experiments have confirmed the contribution of the method to the overall system performance.

The methods provide many opportunities for future development. First of all, they can be generalized for an arbitrary event-based system. Next, the limiting requirement to trigger primary event by primary event to distinguish their impacts can be eliminated by an advanced method for estimating impacts of individual tasks that are influenced by other tasks statistically from multiple observations. Currently, the loading matrix is estimated at the beginning, since it is assumed that its values do not change in time. However, this assumption can be in some cases difficult to fulfill and an adaptation mechanism for the loading matrix might be requested in the future.

Acknowledgements

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS12/188/OHK3/3T/13; and by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth, and the National Foundation for Research, Technology and Development – Austria.

References

[1] M. Pěchouček, S. Thompson, J. Baxter, G. Horn, K. Kok, C. Warmer, R. Kamphuis, V. Mařík, P. Vrba, K. Hall, F. Maturana, K. Dorer, M. Calisti, Agents in industry: the best from the AAMAS 2005 industry track, IEEE Transactions on Intelligent Systems, 21(2), 86 (2006). DOI 10.1109/MIS.2006.19J.

[2] P. Vrba, V. Mařík, P. Siano, P. Leitao, G. Zhabelova, V. Vyatkin, T. Strasser, A Review of Agent and Service-oriented Concepts applied to Intelligent Energy Systems, IEEE Transactions on Industrial Informatics (99), 1 (2014). DOI 10.1109/TII.2014.2326411

[3] O. Yildirim, G. Kardas, A multi-agent system for minimizing energy costs in cement production, Computers in Industry 65(7), 1076 (2014). DOI 10.1016/j.compind.2014.05.002

[4] P. Kadera, P. Tichý, Chilled water system control, simulation, and visualization using Java multi-agent systém, Information Control Problems in Manufacturing, vol. 13 (2009), vol. 13, pp. 1808-1813

[5] R.G. Smith, The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, IEEE Transactions on Computers (12), 1104 (1980)

[6] P. Kadera, P. Tichy, Plan, commit, execute protocol in multi-agent systems, Holonic and multi-agent systems for manufacturing (Springer, 2009), pp. 155 – 164

[7] H.V. Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference architecture for holonic manufacturing systems: PROSA, Computers in Industry 37(3), 255 (1998). DOI 10.1016/S0166-3615(98)00102-X

[8] P. Leitao, F. Restivo, ADACOR: A holonic architecture for agile and adaptive manufacturing control, Computers in Industry 57(2), 121 (2006). DOI 10.1016/j.compind.2005.05.005

[9] A. Giret, V. Botti, Engineering Holonic Manufacturing Systems, Computers in Industry 60(6), 428 (2009). DOI 10.1016/j.compind.2009.02.007

[10] M. Wooldridge, M. Fisher, M.P. Huget, S. Parsons, Model checking multi-agent systems with MABLE, in Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2 (ACM, 2002), pp. 952-959

[11] S. Schliecker, J. Rox, M. Negrean, K. Richter, M. Jersak, R. Ernst, System Level Performance Analysis for Real-Time Automotive Multicore and Network Architectures, IEEE Transactions on Computer-Aided Design of integrated Circuits and Systems, 28(7), 979 (2009)

[12] K. Richter, M. Jersak, R. Ernst, A formal approach to MpSoC performance verification, Computer 36(4), 60 (2003)

[13] R. L. Cruz, A calculus for network delay, IEEE Transactions on Information Theory, 37(1), 114 (1991)

[14] L. Thiele, S. Chakraborty, M. Naedele, Real-time calculus for scheduling hard real-time systems, in Proceedings of IEEE International Symposium on Circuits and Systems, vol. 4 (2000), pp. 101-104

[15] P. Leitao, N. Rodrigues, Modelling and validating the multi-agent system behaviour for a washing machine production line, in Proceedings of IEEE International Symposium on Industrial Electronics (ISIE) (2012), pp. 1203-1208. DOI 10.1109/ISIE.2012.6237260

[16] J. Kubalík, P. Tichý, R. Šindelář, R.J. Staron, Clustering Methods for Agent Distribution Optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 40(1), 78 (2010)

[17] FIPA. Fipa ACL message structure specification (2002)

[18] P.J. Denning, J.P. Buzen, The operational analysis of queueing network models, ACM Computing Surveys (CSUR) 10(3), 225 (1978)

[19] V. Cortellesa, A. D. Marco, P. Inverardi, Model-based software performance analysis, Springer, 2011

[20] G. Casale, G. Serazzi, Bottlenecks identification in multiclass queueing networks using convex polytopes, in Proceedings of The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. pp. 223-230.