2014 Second International Conference on Artificial Intelligence, Modelling and Simulation

Architecture of Real-Time Database in Cloud Environment for Distributed Systems

Sebastijan Stoja
Schneider Electric DMS NS
Novi Sad, Serbia
e-mail: sebastijan.stoja@schneider-electric-dms.com

Bojan Jelačić
Schneider Electric DMS NS
Novi Sad, Serbia
e-mail: bojan.jelacic@schneider-electric-dms.com

Srđan Vukmirović
Computing and Control Department
University of Novi Sad
Novi Sad, Serbia
e-mail: srdjanvu@uns.ac.rs

Darko Čapko
Computing and Control Department
University of Novi Sad
Novi Sad, Serbia
e-mail: dcapko@uns.ac.rs

Nikola Dalčeković
Schneider Electric DMS NS
Novi Sad, Serbia
e-mail: nikola.dalcekovic@schneider-electric-dms.com

***Abstract*— In every distributed system there is a need for a real-time database when working with a large amount of data and when quick response from a service is expected on the client side. Cloud companies do not offer real-time databases, therefore, in this paper we present the architecture of a real-time database in cloud environment, as well as reviews of other papers dealing with this topic, analyzing their advantages and disadvantages. The above-mentioned real-time database attempts to store different kind of data, and our proposed solution is implemented in each distributed system service. Unlike other solutions, this real-time database will not occupy main memory-base because it is running in cloud environment. Also, we are proposing which cloud environment services can be used to implement this real-time database. As an example, we are modeling one distributed system using Petri nets where solution can be applied.**

Keywords-real-time database; cloud computing; distributed systems

I. INTRODUCTION

Companies that use cloud computing have resolved problems like: equipment, licensing, buying large servers, hardware and all of this can be returned back when is no longer needed. Analysts have discovered that computers are used in only 17% of the cases, which represents a great loss for the business. Necessary savings, while improving performance are possible using cloud computing, an option that is becoming increasingly popular in the IT world. Distributed systems in cloud environments are increasingly common due to a number of advantages and novelty of this approach. In modern distributed systems working with a large amount of data, there is a need for a real-time database if quick response from each service is expected. In this paper we present the architecture of real-time database system in cloud environment which is used for applications in distributed systems.

One of the primary reason why this type of in-memory real-time database has been avoided is that it takes a long time to access the data. Recently, in cloud computing, there is no need for a real-time database and applications in distributed systems over the world more and more are moving their solution in cloud environment. The in-memory real-time database described in this paper is used in services to locally store the required data and to allow its access to other services.

This paper is organized as follows: Section I introduced the topic, Section II examines available solutions for real-time databases in the literature, and compares them with our proposal. Section III describes real-time database and transactions in real-time database. In section IV the term cloud computing is described together with its benefits and what offers. In section V we are proposing the architecture of a real-time database. Section VI illustrates one example of using this proposed solution in distributed system and at the end in section VII conclusion of this paper

II. RELATED WORK

There are several existing solutions in the literature, and we will mention the following: The Beehive real-time database [1] which supports advanced real-time transaction scheduling based not on transaction but on data deadline. This real-time database is using to access audio, video, and image data.

The second solution is leaning on message-based approach to prototype the distributed real-time database systems. It is similar to our solution in that it uses client/server paradigm for process interaction [2], however, is does not operate in cloud environment and requires strong servers with good performances due to substantial memory consumption.

Paper [3] presents a centralized and decentralized database model using Petri nets in which they illustrate

978-1-4799-7600-3/14 $31.00 © 2014 IEEE
DOI 10.1109/AIMS.2014.54

reachability graph of the net but the state of the model is not analyzed. That leaves questions like whether the Petri net is alive, if agents fall into deadlocks, if tokens are bounded, etc. unexplained.

All implemented solutions [1][2][4][5][6][7][8] take main memory-based real-time database system approaches which limits their application due to memory consumption. Unlike these approaches, our proposed solution runs in cloud environment, which will not take memory and thus increases the speed of the server response.

III. REAL-TIME DATABASE

Computers are the most important part of real-time systems, therefore real-time computing is an important discipline in computer science and engineering [9].

In recent years, the importance of real-time systems has been growing, which is a result of the increasing amount of data and requirements for durability, security and consistency of the processed data. As real-time systems [21][22] evolve, their applications become more complex and require access to more data. Such requirements are called database systems. Furthermore, merging database and real-time technology represents an integrated system, which provides database operations with real-time constraints and these are generally called real-time database systems. Like a conventional database system, a real-time database system functions as a repository of data, provides efficient storage, and performs retrieval and manipulation of information. However, as a part of a real-time system that are characterized by time constraints, a real-time database systems has the added burden of ensuring some degree of confidence in meeting the system's timing requirements [10]. These systems (real-time database systems) are used in applications which are running in real-time, in applications for distributed systems where write/read speed is crucial due to real-time processing.

A. Real-time transactions

A typical real-time system consists of several transactions that must be executed concurrently. Each transactions has a value, which is gained to the system if a computation finishes in a specific time. Each transaction also has a deadline, which indicates a time limit, when a result of the computing becomes useless [11].

A real-time transaction is a transaction with additional real-time attributes. There is an additional attribute for a real time transaction. These attributes are used by the real-time scheduling algorithm and concurrency control method. The additional attributes are the following [10]:

- Timing constraints - e.g. deadline is a timing constraint associated with the transaction.
- Criticalness - It measures how critical it is that a transaction meets its timing constraints. Different transactions have different criticalness. Furthermore, criticalness is a different concept from deadline because a transaction may have a very tight deadline but missing it may not cause great harm to the system.
- Value function - is related to a transaction's criticalness. It measures how valuable it is to complete the transaction at some point in time after the transaction arrives.
- Resource requirements - Indicates the number of I/O operations to be executed, expected CPU usage, etc.
- Expected execution time. Generally very hard to predict but can be based on estimate or experimentally measured value of worst case execution time.
- Data requirements - Read sets and write sets of transactions.
- Periodicity - If a transaction is periodic, what its period is.
- Time of occurrence of events - In which point in time a transaction issues a read or write request.
- Other semantics - Transaction type (read-only, write-only, etc.).

The real-time database system apply all three types of transactions:

- Write-only transactions obtain the state of the environment and write into the database.
- Update transactions derive a new data item and store it in the database.
- Read-only transactions read data from the database and transmit that data or derived actions based on that data to the controlling system.

IV. CLOUD COMPUTING

Cloud computing [12][13][14] is the most important technology in computing industry right now. It is similar like electricity grid over a network. Cloud computing allows users to migrate and from any physical location to access their data. It provides following key characteristics: agility, application programming interface (API), cost, device and location independence, maintenance, multitenancy, performance, productivity, reliability, scalability and elasticity and security.

Some benefits using cloud computing in distributed systems:

- Unlimited resources: platform provides users that has unlimited resources. The user does not need to worry about the potential lack of processing power and limited storage space.
- Minimization of infrastructure risk: using cloud reduces the risk that exists when purchasing the necessary computer infrastructure and allows users to easily expand and decrease depending on the needs of the application.
- Scalability: enabled easy expandability amount of resources needed for the execution of the client application in order to handling a larger number of users or remove unnecessary resources in case of a small number of users.
- Pay as you go model: unlike traditional solutions, working with cloud does not require any start-up costs thanks to the scalability of the platform and can

be achieved the maximum level of hardware efficiency and usability could be achieved.

Many cloud companies in the type of services offer three models:

- IaaS – is a model that allows to rent hardware such as servers, network technology, data storage and computer centers as a service.
- PaaS – is a model which provides an environment with defined programming languages and application interface to support all phases of application development from planning, design and development to testing.
- SaaS – is a model which provides the ability to access applications without requiring them to be installed and run on its own computer.

Cloud services can be private, hybrid and public. The standard of public cloud services offer their service dependent on selected models of service resources from public data centers. These services do not guarantee user privacy in terms of installed hardware or software, as opposed to private services where clients install the required hardware and software that is used by registered users only. In fact, private cloud computing means that cloud architecture is located behind a firewall of organizations and provides IT services for internal guide. Hybrid services offer a combination of the two mentioned above, where one part, not critical for the business, is held in a public data center and the other part in a company.

Public cloud computing services are directed to consumer services such as: searching on the internet, personal email services (Yahoo, Gmail, and Hotmail), social networking, etc. Small and beginner companies attract concept of services provided by public cloud computing because it allows the reduction of the initial investment in IT equipment. However, for many large companies, IT infrastructure is closely associated with the central area of their operations and outsourcing of computer capacity would represent a major business risk. Safety, reliability, performance and compliance with standards are the most important issues for managers of IT departments when evaluating new technologies. Such companies can develop an internal, private cloud computing network over which they will have a greater control and achievement of security in computer resources.

Companies present in the market which offer their cloud offering among others are Microsoft (Windows Azure), Google (App Engine), IBM (Smart Cloud), and Amazon (Elastic Compute Cloud).

V. ARCHITECTURE DESIGN

In this section is presented the structure of one distributed system [15] in cloud environment that uses in-memory real-time database. From the Fig. 1, we can see that this system contains clients which are using services that are running in cloud environment. Client #1 communicates with services from first row (Service #1, Service #2, Service #n), Client #2 communicates with services from the second row, etc.

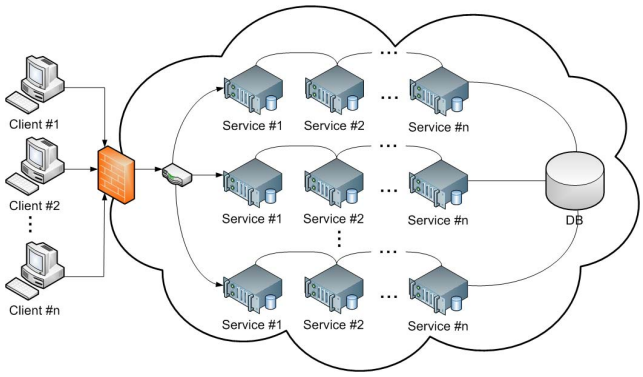

Figure 1. Architecture of distributed system in cloud environment.

An in-memory real-time database is placed inside of each service, so each service contains own in-memory real-time database and she owns own data.

Such architecture is used because clients' side needs quick response from a service. Only with in-memory real-time database distributed systems can work rapidly and they will have faster response from a service. All services have mutual communication among themselves and through that communication they exchange data. In addition, communications between services and between client and services has to be very quick that it would go unnoticed on the client's side. Such communication of exchanging data is made through mechanism called Publisher and Subscriber [16] which is of great importance for distributed systems.

This in-memory real-time database of an each service it has possibility to store data in a cloud database. Storing data is most frequently doing in the following situations: when service is shutting down, when service failure or depends on mode (it can be configured) of a service which can store for example once a day/ week, etc.

This in-memory real-time database is designed to be modular. Most of the modules are optional. Modules can be enabled/disabled depending on the requirements of the service, which utilize the in-memory real-time database.

To support transactional behavior, in-memory real-time database uses the command-pattern to implement operations. This way bulk operations can be committed or rolled back, if one of the operations failed.

In contrast with other solutions of real-time database, our solution does not use computer memory because every operation is done in a cloud environment and this represents a major advantage.

In Fig. 2, the following concepts are represented: communication from client to an in-memory real-time database located in a service, communication in in-memory real-time database, making query and types of operations. The algorithm on Fig. 2 shows that the client first makes a transaction or send query to get data from the in-memory real-time database located in service. The query first arrives in Message identification where it is investigated and validated. Validation works as follows: first, the authenticity of the query is checked. If false, an empty query is sent back to the client. If the message is valid, next step is Message identification. This in-memory real-time database supports

three types of message: add, update and delete. This allows Message identification to also take care of the query identification.

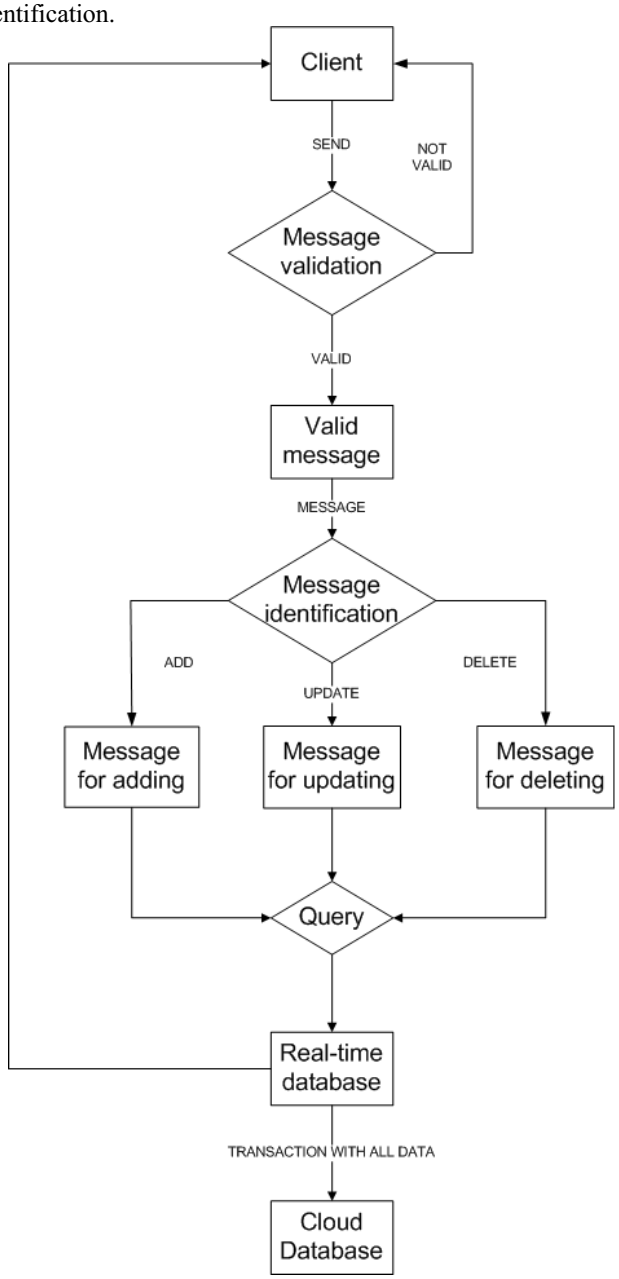

Figure 2. Algorithm of real-time database.

Next, depends the type of message, there are three possibilities for further action: Message for adding, Message for updating and Message for deleting. Through one of these three branches, a query is prepared based on the message type, which is then saved in in-memory real-time database. If an error occurs during the access to the database, a rollback will turn on and all data will be back to the initial state (prior to database access). With this module, the loss of data that may occur during the transaction is avoided. At the end, the query from in-memory real-time database is sent to the client. As mentioned above this transaction of sending data is done with mechanism Publisher/Subscriber. When the service is shutting down, all data from real-time database services will be packed and sent in one query to a cloud database where it will be stored on existing data.

Implementing proposed solution for cloud environment can be used by companies that offer Platform as a Service, and these are:

- Microsoft Windows Azure
  - Azure Worker role which is a cloud service and services can be run on it,
  - Azure cache which is distributed, in-memory, scalable solutions that enable you to build highly scalable and responsive applications by providing super-fast access the data, and
  - Azure SQL Database is relational database-as-a-service that delivers predictable performance, scalability, business continuity, data protection, and near-zero administration to cloud developers and solution architects.
- Amazon Elastic Compute Cloud
  - Services should be run in App Services which is a fully-managed service in the AWS Cloud that makes it simple and cost-effective to set up, manage, and scale a custom search solution for application,
  - Elastic Cache it can scale an in-memory cache in the cloud, and
  - Amazon RDS which is a web service that makes it easy to set up, operate, and scale a relational database in the cloud.
- Or instead of using database from Azure and Amazon, a cloud database can be designed [16] and installed on servers.

VI. EXAMPLE USING REAL-TIME DATABASE IN DISTRIBUTED SYSTEMS

On Fig. 3 is shown an example using real-time database system in on distributed system.

Real-time database is located in each service that handles different data. The meaning of each service and its type of data is described below.

SSE - Service for static elements. In the real-time database located in this service is stored only data of a model that describes the static elements of the distributed system - a network model which is based on CIM [18], but has been adapted for use in power applications.

SDE - Service for dynamic elements. In this real-time database is stored current dynamic state of the device (state switchgear) in the electric power distribution system.

TS - Topology service. This real-time database located in this service stores the representation (topology) of the distributed system, which is based on the current state of the switchgear equipment.

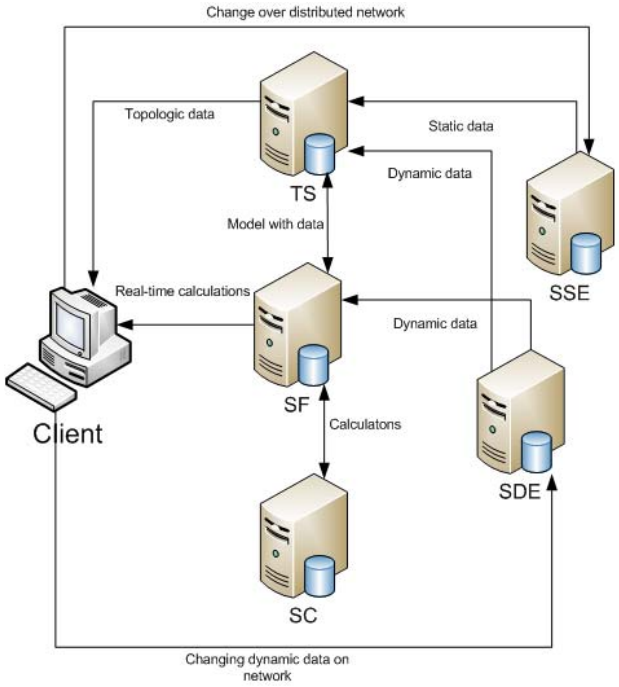

Figure 3. Example of using real-time database in distributed system.

SF - Service for functions. The Real-time database in this service is responsible for executing real-time calculations, which are triggered by the changes in other services.

SC - Service calculations. As the name suggests, this service real-time database stores data of calculations.

Client - Making interactions including the review of the current state of the device, control action, presenting and processing of alarm confirmation, trending, etc.

During modeling a distributed system uses Petri nets [19][20] with next case: in the system, the final numbers of objects (static elements) are brought through SSE service, while SDE is ignored and linked with other services.

When transforming distributed system into Petri net form:

$$N_1 = (P,T,A,W,M_0). \quad (1)$$

Symbols from Fig. 4 have following meaning:

- p_x - static elements
- p_1- model with static elements
- p_2 - model with data
- p_3 - job for calculation
- p_4 - result of calculations
- p_5 - topologic data
- p_6 - real-time calculations

Following at Fig. 4, reachability graph and reachability three can be obtained, and it can be concluded that this system is finite, the number of tokens of every place in all markings is bounded, there are not dead transitions, and it is a live system - meaning that the interaction between the agents does not fall into deadlocks.

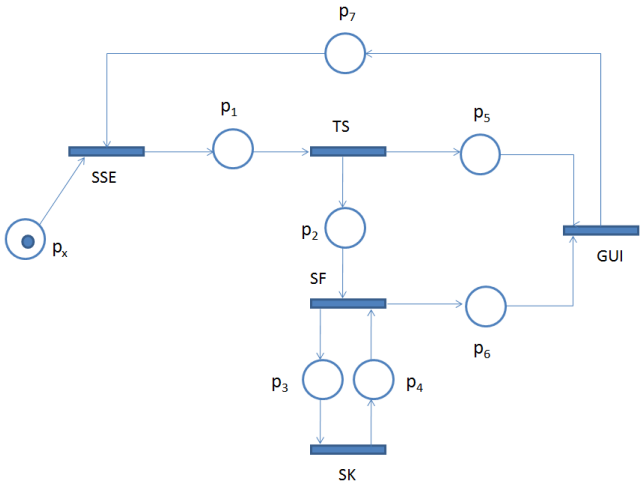

Figure 4. Distributed system modeled with Petri net.

VII. CONCLUSION

The aim of this paper was to present the architecture of a real-time database used in distributed systems. In this research, better results are given when a solution is implemented in each service of a distributed system that works with different kinds of data. It is therefore truly inferred that such a solution will give better performance without overloading performances of the machine because of it being applied in a cloud environment. The only deficiency of this solution is that cloud technology has not been sufficiently improved yet and access to services in the cloud takes time, which is the main reason why the system can be slow. This paper also discusses an example of a distributed system which is modeled by Petri nets, which shows in that the distributed system is a live, bounded, and finite and never falls into deadlocks.

For further research, this solution will be implemented on one of cloud platforms, it will measure performances and compare with other solutions of real-time database in and out of the cloud environment.

REFERENCES

[1] J.A. Stankovic, S.H. Son, and J. Liebeherr, “Beehive: Global Multimedia Database Support for Dependable, Real-Time Applications,” Proc. Second Int’l Workshop Active, Real-Time, and Temporal Database Systems, 1998, pp. 51-72

[2] Son, S.H., " An environment for prototyping real-time distributed databases", Proceedings of the First International Conference on Systems Integration, 1990. Systems Integration '90., Apr. 1990, pp.358-367

[3] Fricks, R.M., Puliafito, A. ; Trivedi, K.S., " Performance analysis of distributed real-time databases ", Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International, Sep. 1998, pp.184-194

[4] K.-D. Kang, J. Oh, and S.H. Son, “Chronos: Feedback Control of a Real Database System Performance” Proc. 28th IEEE Int’l Real-Time Systems Symp. (RTSS), Dec. 2007, pp.267-276

[5] B. Adelberg, “STRIP: A Soft Real-Time Main Memory Database for Open Systems”, PhD dissertation, Stanford Univ., 1997.

[6] C.-S. Peng, K.-J. Lin, C. Boettcher, “Real-Time Database Benchmark Design for Avionics Systems” Proc. First Int’l Workshop Real-Time Databases: Issues and Applications (RTDB), 1997, pp.123-138

[7] Chenggang Zhen, Kai Li, "Memory management research based on real-time database", Test and Measurement, 2009. ICTM '09. International Conference on (Volume:1), Dec. 2009, pp.416-419

[8] Jianming Qiu, Jizhen Liu, Yuebin Hou, Jianhua Zhang, "Use of real-time/historical database in Smart Grid", International Conference on Electric Information and Control Engineering (ICEICE), Apr. 2011, pp. 1883 - 1886

[9] Shin, Kang G, "Introduction to the Special Issue on Real-Time Systems", Computers, IEEE Transactions on (Volume:C-36,Issue: 8), Aug. 1987, pp.901-902J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73

[10] B. Kao and H. Garcia-Molina. An overview of real-time database systems. In S. H. Son, editor, Advances in Real-Time Systems, Prentice Hall, 1995, pp.463–486

[11] Fernandes, Y.M.P, Perkusich, A. ; Neto, P.F.R. ; Perkusich, M.L.B, "Implementation of transactions scheduling for real-time database management", Systems, Man and Cybernetics, 2004 IEEE International Conference on (Volume:6), Oct. 2014, pp. 5136 - 5141 vol.6

[12] Simon Bradshaw, Christopher Millard, Ian Walden, "Contracts of Cloud: Comparison and Analysis of the Terms and Conditions of Cloud Computing Service", Queen Mary School of Law Legal Studies Research Paper No. 63/2010

[13] Lamia Youseff, Maria Butrico, Dilma Da Silva, Toward a Unified Ontology of Cloud Computing, Grid Computing Environments Workshop, 2008. GCE '08 New York, Nov. 2008, pp. 1-10

[14] Radu Prodan and Simon Osterman, A Survey and Taxonomy of Infrastructure as a Service and Web Hosting Cloud Providers, Grid Computing, 2009 10th IEEE/ACM International Conference Austria, Oct.2009, pp. 17-25

[15] Andrew S.Tanenbaum, Maarten Van Steen, Distributed Systems Second Edition, Pearson Education, 2007J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73

[16] Stoja S., Vukmirovic S., Jelacic B., "Publisher/Subscriber Implementation in Cloud Environment", International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Oct.2013, pp.677-682

[17] Ferretti, L., Pierazzi, F., Colajanni, M., Marchetti, M., "Performance and Cost Evaluation of an Adaptive Encryption Architecture for Cloud Databases", Cloud Computing, IEEE Transactions on (Volume:2 , Issue: 2), Apr.-June 2014, pp.143-155

[18] Alan W. McMorran, "An Introduction to IEC 61970-301 & 61968-11: The Common Information Model", Univetsity of Strathclyde, Glasgow UK, 2007

[19] T. Murata, “Petri nets: Properties, analysis and applications, ”Proceedings of the IEEE, vol. 77, no. 4, April 1989, pp. 541–580

[20] Celaya, J.R.,Desrochers, A.A.; Graves, R.J.,“Modeling and analysis of multi-agent systems using petri nets“,IEEE Transactions on Systems, Man and Cybernetics, October 2007, pp.1439-14

[21] Pelusi D., Mascella R., "Optimal control algorithms for second order systems", Journal of Computer Science, 9 (2), 2013, pp. 183-197

[22] Pelusi D., "Improving settling and rise times of controllers via intelligent algorithms", 14th International Conference on Modelling and Simulation, UKSim, 2012, pp. 187-192