FUTURE COMPUTING 2018

The Tenth International Conference on Future Computational Technologies and Applications

ISBN: 978-1-61208-608-8

February 18 - 22, 2018

Barcelona, Spain

FUTURE COMPUTING 2018 Editors

Eugen Borcoci, University Politehnica of Bucharest, Romania
Claus-Peter Rückemann, Leibniz Universität Hannover /
Westfälische Wilhelms-Universität Münster /
North-German Supercomputing Alliance (HLRN), Germany

FUTURE COMPUTING 2018

Forward

The Tenth International Conference on Future Computational Technologies and Applications (FUTURE COMPUTING 2018), held between February 18 - 22, 2018 - Barcelona, Spain, continued a series of events targeting advanced computational paradigms and their applications. The target was to cover (i) the advanced research on computational techniques that apply the newest human-like decisions, and (ii) applications on various domains. The new development led to special computational facets on mechanism-oriented computing, large-scale computing and technology-oriented computing. They are largely expected to play an important role in cloud systems, on-demand services, autonomic systems, and pervasive applications and services.

The conference had the following tracks:

- Computing technologies
- Computational intelligence strategies
- Challenges

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the FUTURE COMPUTING 2018 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to FUTURE COMPUTING 2018. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the FUTURE COMPUTING 2018 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope FUTURE COMPUTING 2018 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of future computational technologies and applications. We also hope that Barcelona provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

FUTURE COMPUTING 2018 Chairs

FUTURE COMPUTING Steering Committee

Hiroyuki Sato, The University of Tokyo, Japan
Kendall E. Nygard, North Dakota State University - Fargo, USA
Alex Wijesinha, Towson University, USA
Albert Zomaya, University of Sydney, Australia
Sergio Ilarri, University of Zaragoza, Spain
Dan Tamir, Texas State University, USA
Wail Mardini, Jordan University of Science and Technology, Jordan

FUTURE COMPUTING 2018 Industry/Research Advisory Committee

Francesc Guim, Intel Corporation, Spain
Yasushi Kambayashi, Nippon Institute of Technology, Japan
Jay Lofstead, Sandia National Laboratories, USA

FUTURE COMPUTING 2018

Committee

FUTURE COMPUTING Steering Committee

Hiroyuki Sato, The University of Tokyo, Japan
Kendall E. Nygard, North Dakota State University - Fargo, USA
Alex Wijesinha, Towson University, USA
Albert Zomaya, University of Sydney, Australia
Sergio Ilarri, University of Zaragoza, Spain
Dan Tamir, Texas State University, USA
Wail Mardini, Jordan University of Science and Technology, Jordan

FUTURE COMPUTING 2018 Industry/Research Advisory Committee

Francesc Guim, Intel Corporation, Spain
Yasushi Kambayashi, Nippon Institute of Technology, Japan
Jay Lofstead, Sandia National Laboratories, USA

FUTURE COMPUTING 2018 Technical Program Committee

Andrew Adamatzky, University of the West of England, Bristol, UK
Panagiotis D. Bamidis, Aristotle University of Thessaloniki, Greece
Rudolf Berrendorf, Bonn-Rhein-Sieg University of Applied Sciences Sankt Augustin, Germany
Abdelhani Boukrouche, University of Guelma, Algeria
Christos Bouras, University of Patras | Computer Technology Institute & Press <Diophantus> Greece
Massimiliano Caramia, University of Rome "Tor Vergata", Italy
Chin-Chen Chang, Feng Chia University, Taiwan
Pabitra Pal Choudhury, International Statistical Educational Center (ISEC) | Indian Statistical Institute, Kolkata, India
Isabel Maria de Sousa de Jesus, ISEP-Institute of Engineering of Porto, Portugal
Leandro Dias da Silva, Instituto de Computação | Universidade Federal de Alagoas, Brazil
Qin Ding, East Carolina University, USA
Sandro Fonseca de Souza, Universidade do Estado do Rio de Janeiro, Brazil / CERN - European Laboratory for Particle Physics, Swizerland
Francesco Fontanella, Università di Cassino e del Lazio Meridionale, Italy
Félix J. García Clemente, University of Murcia, Spain
J Paul Gibson, Telecom Sud Paris, France
Apostolos Gkamas, University Ecclesiastical Academy of Vella of Ioannina, Greece
Victor Govindaswamy, Concordia University - Chicago, USA
George A. Gravvanis, Democritus University of Thrace, Greece
David Greenhalgh, University of Strathclyde, Scotland
Francesc Guim, Intel Corporation, Spain
Houcine Hassan, Universitat Politècnica de València, Spain

Tzung-Pei Hong, National University of Kaohsiung, Taiwan
Wei-Chiang Hong, Jiangsu Normal University, China
Andrei Hossu, University Politehnica of Bucharest, Romania
Daniela Hossu, University Politehnica of Bucharest, Romania
Woomin Hwang, National Security Research Institute, South Korea
Sergio Ilarri, University of Zaragoza, Spain
Yuji Iwahori, Chubu University, Japan
Yasushi Kambayashi, Nippon Institute of Technology, Japan
Martin Klimo, University of Zilina, Slovakia
Michihiro Koibuchi, National Institute of Informatics, Japan
Zbigniew Kokosinski, Cracow University of Technology, Poland
Piotr A. Kowalski, System Research Institute - Polish Academy of Sciences / AGH University of Science and Technology, Poland
Marek Kretowski, Bialystok University of Technology, Poland
Carlos Leon, Universidad de Sevilla, Spain
Vitus Leung, Sandia National Laboratories, USA
Stan Z. Li, Institute of Automation - Chinese Academy of Sciences, China
Lan Lin, Ball State University, USA
An-An Liu, Tianjin University, China
Jay Lofstead, Sandia National Laboratories, USA
Szymon Lukasik, AGH University of Science and Technology / System Research Institute - Polish Academy of Sciences, Poland
Khoa Luu, Carnegie Mellon University, USA
Stephane Maag, Telecom SudParis, France
Olaf Maennel, Tallinn University of Technology, Estonia
Giuseppe Mangioni, DIEEI - University of Catania, Italy
Francesco Marcelloni, University of Pisa, Italy
Wail Mardini, Jordan University of Science and Technology, Jordan
Marjan Mernik, University of Maribor, Slovenia
Yehya Mohamad, Fraunhofer FIT, Germany
Isabel Muench, German Federal Office for Information Security, Germany
Anand Nayyar, KCL Institute of Management and Technology, Jalandhar, Punjab, India
Vilem Novak, University of Ostrava, Czech Republic
Kendall E. Nygard, North Dakota State University, USA
Gonzalo Pajares, University Complutense of Madrid, Spain
Sankar K. Pal, Indian Statistical Institute, India
Linqiang Pan, Huazhong University of Science and Technology, China
Frederick Petry, Naval Research Laboratory, USA
Prakash Ranganathan, University of North Dakota, USA
Carsten Röcker, Fraunhofer IOSB-INA, Germany
Sandeep Sagar K.S., UVCE | K.R. Circle, Bangalore, India
Alessia Saggese, University of Salerno, Italy
Božidar Šarler, University of Nova Gorica | Institute of Metals and Technology, Slovenia
Hiroyuki Sato, The University of Tokyo, Japan
Peter Schartner, Klagenfurt University - System Security Group, Austria
Elad Michael Schiller, Chalmers University of Technology, Sweden
Andrew Schumann, University of Information Technology and Management in Rzeszow, Poland
Friedhelm Schwenker, University of Ulm, Germany

Table of Contents

A Novel Training Algorithm based on Limited-Memory quasi-Newton Method with Nesterov's Accelerated Gradient for Neural Networks 1
Shahrzad Mahboubi and Hiroshi Ninomiya

Four-State Partial Synchronizers for a Large-Scale of Processors -Symmetric Synchronizers 4
Hiroshi Umeo and Naoki Kamikawa

Energy Efficient Real-time Scheduling Algorithm for Microcontroller Units 10
Douglas Lautner, Xiayu Hua, Miao Song, Scott DeBates, and Shangping Ren

Representing Medication Guidelines for Use in Production Rule Systems in the Context of the POLYCARE Project 15
Jonas Konning, Carlos Velasco, Yehya Mohamad, Stefan Decker, and Oya Beyan

Edge Computing Architectures - A Survey on Convergence of Solutions 23
Eugen Borcoci and Serban Obreja

A Novel Training Algorithm based on Limited-Memory quasi-Newton Method with Nesterov's Accelerated Gradient for Neural Networks

Shahrzad Mahboubi and Hiroshi Ninomiya
Department of Information Science, Shonan Institute of Technology
Email: mahboubi.shaa@gmail.com, ninomiya@info.shonan-it.ac.jp

***Abstract*—This paper describes a novel training algorithm based on Limited-memory quasi-Newton method (LQN) with Nesterov's accelerated gradient for faster training of neural networks.**

Keywords–Limited-memory quasi-Newton method; Nesterov's accelerated gradient method; neural networks; training algorithm.

I. Introduction

Neural networks have been recognized as a useful tool for the function approximation problems with high-nonlinearity [1][2]. Training is the most important step in developing a neural network model. Gradient algorithm based on the first order approximation such as Steepest gradient (SG), Momentum and Nesterov's accelerated gradient (NAG) methods are popularity used for this purpose [1]-[3]. With the progress of AI technologies the characteristics between inputs and desired outputs of the training samples become increasingly complex. Then neural networks have to train the highly-nonlinear functions. Under such circumstances the first order methods converge too slowly and optimization error cannot be effectively reduced within finite time in spite of its advantage [2]. The quasi-Newton (QN) training, which is one of the most effective optimization based on the second order approximation [4] is widely utilized as the robust training algorithm for highly-nonlinear function approximation. However, the QN iteration includes the approximated Hessian, that is QN needs the massive computer resources of memories as the scale of neural network becomes larger. To deal with this problem, it is noteworthy that QN incorporating Limited-memory scheme is effective for large-scale problems.

In this paper, the acceleration technique of Limited-memory QN (LQN) is proposed using Nesterov's accelerated gradient. In [2], the QN training was drastically accelerated by Nesterov's accelerated gradient that is called Nesterov's accelerated quasi-Newton (NAQ) training. Therefore, the proposed algorithm can be accelerated in cooperating the similar scheme of NAQ into LQN. The method is referred to as Limited-memory NAQ (LNAQ). The proposed algorithm is demonstrated through the computer simulations for a benchmark problem compared with the conventional training methods.

II. Formulation of training and Limited-memory quasi-Newton method

This section describes an error function of the neural network training and conventional training algorithms beased on gradients such as Back propagetion and Limited-memory quasi-Newton method.

A. Formulation of training

Let $\mathbf{d}_p$, $\mathbf{o}_p$ and $\mathbf{w} \in \mathbb{R}^D$ be the p-th desired, output, and weight vectors, respectively the error function $E(\mathbf{w})$ is defined as the mean squared error (MSE) of

$$E(\mathbf{w}) = \frac{1}{|T_r|}\sum_{p\in T_r} E_p(\mathbf{w}),\quad E_p(\mathbf{w}) = \frac{1}{2}\|\mathbf{d}_p - \mathbf{o}_p\|^2, \tag{1}$$

where T_r denotes a training data set $\{\mathbf{x}_p, \mathbf{d}_p\}, p \in T_r$ and $|T_r|$ is the number of training samples. Among the gradient-based algorithms, (1) is minimized by $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$, where k is the iteration count and $\mathbf{v}_{k+1}$ is the update vector. SG, so-called Back propagation method has $\mathbf{v}_{k+1} = -\alpha_k \nabla E(\mathbf{w}_k)$ with the step size α_k and the gradient vector at $\mathbf{w}_k$ of $\nabla E(\mathbf{w}_k)$.

B. Limited-memory quasi-Newton training (LQN)

The update vector of QN is defined as

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k). \tag{2}$$

where $\mathbf{H}_k$ is the symmetric positive definite matrix and iteratively approximated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [4]. For the purpose of reducing the amount of memory used in QN a sophisticated technique incorporating the limited-memory scheme is widely utilized for the calculation of $\mathbf{v}_{k+1}$ of LQN. Specifically, this method is useful for solving problems whose $\mathbf{H}_k$ (inverse of approximated Hessian) matrices in (2) cannot be computed at a reasonable cost [4]. Furthermore, instead of storing $D \times D$ matrix of $\mathbf{H}_k$, only $2 \times t$ vectors of the dimension D ($2 \times t \times D$) have to be stored. Here, t is defined by users and $t \ll D$.

III. Proposed algorithm -LNAQ

Nesterov's accelerated quasi-Newton (NAQ) training was derived by the quadratic approximation of (1) around $\mathbf{w}_k + \mu\mathbf{v}_k$, and μ was the momentum coefficient whereas QN used the approximation of (1) around $\mathbf{w}_k$ [2]. NAQ drastically improved the convergence speed of QN using the gradient vector at $\mathbf{w}_k + \mu\mathbf{v}_k$ of $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$ called Nesterov's accelerated gradient vector [3]. In this paper, we apply the limited-memory method into NAQ, that is called LNAQ. The proposed method can be expected to cope with large-scale optimization problems like LQN, maintaining the fast training of NAQ. The update vector of NAQ is

$$\mathbf{v}_{k+1} = \mu\mathbf{v}_k - \alpha_k \hat{\mathbf{H}}_k \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k), \tag{3}$$

 ISBN: 978-1-61208-608-8

and the matrix $\hat{\mathbf{H}}_k$ was updated by,

$$\hat{\mathbf{H}}_{k+1} = \hat{\mathbf{H}}_k - \frac{(\hat{\mathbf{H}}_k\mathbf{q}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k(\hat{\mathbf{H}}_k\mathbf{q}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k} + \left(1 + \frac{\mathbf{q}_k^{\mathrm{T}}\hat{\mathbf{H}}_k\mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right)\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}, \quad (4)$$

where

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu\mathbf{v}_k), \quad (5)$$

$$\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k). \quad (6)$$

(4) was equivalent to the BFGS formula of [4] by replacing $\mathbf{p}_k$ and $\mathbf{q}_k$ into $\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k$ and $\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k)$, respectively. Therefore, the limited-memory scheme can be straightly applied to NAQ as illustrated in Algorithms 1 and 2. In Algorithm 1, two calculations of gradient vectors of $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$ and $\nabla E(\mathbf{w}_{k+1})$ were needed within a training loop whereas LQN was calculated once. This is a disadvantage of LNAQ, but the algorithm can farther shorten the iteration counts to cancel out the effect of this shortcoming.

Algorithm1: The proposed LNAQ

1. $k = 1$;
2. $\mathbf{w}_1 = rand[-0.5, 0.5]$(uniform random numbers);
3. **While**$(E(\mathbf{w}_k) > \varepsilon \text{ and } k < k_{max})$
   (a) Calculate $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$;
   (b) Calculate the direction vector $\hat{\mathbf{c}}_k$ using Algorithm 2;
   (c) Calculate α_k using Armijo's condition;
   (d) Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mu\mathbf{v}_k - \alpha_k\hat{\mathbf{c}}_k$;
   (e) Calculate $\nabla E(\mathbf{w}_{k+1})$;
   (f) $k = k + 1$;
4. **return** $\mathbf{w}_k$;

Algorithm2: Direction Vector of LNAQ

1. $\hat{\mathbf{c}}_k = -\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$;
2. for $i : k, k-1, \ldots, k - \min(k, (t-1))$;
   (a) $\hat{\beta}_i = \mathbf{p}_i^{\mathrm{T}}\hat{\mathbf{c}}_k / \mathbf{p}_i^{\mathrm{T}}\mathbf{q}_i$;
   (b) $\hat{\mathbf{c}}_k = \hat{\mathbf{c}}_k - \hat{\beta}_i\mathbf{q}_i$;
3. if $k > 1, \hat{\mathbf{c}}_k = (\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k / \mathbf{q}_k^{\mathrm{T}}\mathbf{q}_k)\hat{\mathbf{c}}_k$;
4. for $i : k - \min(k, (t-1)), \ldots, k, k-1$;
   (a) $\hat{\tau} = \mathbf{q}_i^{\mathrm{T}}\hat{\mathbf{c}}_k / \mathbf{q}_i^{\mathrm{T}}\mathbf{p}_i$;
   (b) $\hat{\mathbf{c}}_k = \hat{\mathbf{c}}_k - (\hat{\beta}_i - \hat{\tau})\mathbf{p}_i$;
5. **return** $\hat{\mathbf{c}}_k$;

IV. SIMULATION RESULTS

In this paper, we demonstrated the effectiveness of the proposed LNAQ for the training of neural networks. Levy function as shown in (7) is used for the function approximation problem. Levy function is a multimodal function and has the highly-nonlinear characteristic [2].

$$f(x_1 \ldots x_n) = \frac{\pi}{n}\left\{\sum_{i=1}^{n-1}[(x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1}))] + 10\sin^2(\pi x_1) + (x_n - 1)^2\right\}, x_i \in [-4, 4], \forall i. \quad (7)$$

Here the structure of neural network is 5-50-1, that is, the network has 5 inputs, 1 output and 50 hidden neurons. The dimension of $\mathbf{w}$ is 351. The number of training data is $|T_r| = 5000$, which are generated by uniformly random number in $x_i \in [-4, 4]$. The storage amount of t is 30 for LQN and LNAQ. Ten simulation runs are performed from different initial values. The propose LNAQ is compared with NAG, ADAM and LQN. ADAM is one of the latest and the most effective first order training method [5]. The momentum coefficient μ of NAG and LNAQ is experimentally set to 0.95 in the simulations. The termination conditions are set to $E(\mathbf{w}) \leq 10^{-4}$ and $k_{max} = 5\times10^5$. Figure 1 shows the training error $E(\mathbf{w})$ of NAG, ADAM, LQN and LNAQ for the iteration count in the early stage of training. From this figure, it can be seen that NAG cannot converge within 5×10^4 iterations and the iteration counts of LNAQ is drastically improved. Next, the averages of iteration counts and CPU times (sec) for NAG, ADAM, LQN and LNAQ are illustrated in Table 1. In the simulations, the trainings continued until each training error of $E(\mathbf{w})$ was less than 1×10^{-4}. This is an important point of the function approximation problem. That is, the trained network with the small MSE of $E(\mathbf{w})$ can become an accurate neural network model. All algorithms can obtain the neural network model with the small training error. However, the first order methods (NAG and ADAM) need more iteration counts and CPU times than LQN and LNAQ. Furthermore, LNAQ was able to improve the convergence speed compare with LQN. As a result, it is confirmed that the proposed LNAQ is efficient and practical for the training of neural networks.

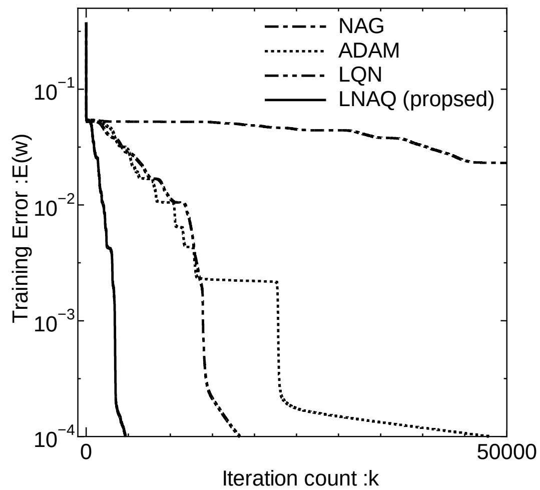


Figure 1. The average training errors for iteration count.

TABLE 1. SIMULATION RESULTS FOR THE AVERAGE ITERATION COUNTS AND CPU TIMES.

Algorithm	NAG	ADAM	LQN	LNAQ
Iteration counts	335,403	48,680	17,892	4,514
CPU times (sec)	1,820	263	142	60

V. CONCLUSIONS

In this research, we proposed a novel training algorithm called LNAQ which was developed based on Limited-memory method of QN using Nesterov's accelerated gradients. The effectiveness of the proposed LNAQ was demonstrated through the computer simulations compared with the conventional algorithms such as NAG, ADAM and LQN. It helps provide accurate neural network models much fast. In the future, the validity of the proposed algorithm for more highly nonlinear function approximation problems and the much huge scale problems including deep networks will be demonstrated.

ACKNOWLEDGMENT

This work was supported by Japan Society for the Promotion of Science (JSPS), KAKENHI (17K00350).

REFERENCES

[1] S. Haykin, Neural Networks and Learning Machines 3^{rd}, Pearson, 2009.

[2] H. Ninomiya, "A novel quasi-Newton-Optimization for neural network training incorporating Nesterov's accelerated gradient", *IEICE NOLTA Journal*, **vol.E**8-**N**, no.4, p.289-301, Oct. 2017.

[3] Y. Nesterov, "*Introductory Lectures on Convex Optimization: A Basic Course*", Kluwer Boston, 2004.

[4] J. Nocedal, and S.J. Wright., *Numerical Optimization Second Edition,* Springer, 2006.

[5] S. Ruder,"An overview of gradient descent optimization algorithms", arXiv 1609.04747, 15. Jun. 2017.

 ISBN: 978-1-61208-608-8

Four-State Partial Synchronizers for a Large-Scale of Processors – Symmetric Synchronizers –

Hiroshi Umeo[†] and Naoki Kamikawa[†]
[†]School of Information Engineering
University of Osaka Electro-Communication
Neyagawa-shi, Hastu-cho, 18-8, Osaka, Japan
corresponding email address: umeo@cyt.osakac.ac.jp

***Abstract*—The synchronization in cellular automata has been known as the Firing Squad Synchronization Problem (FSSP) since its development, where the FSSP gives a finite-state protocol for synchronizing a large scale of cellular automata. A quest for smaller state FSSP solutions has been an interesting problem for a long time. Umeo, Kamikawa and Yunès [2009] answered partially by introducing a concept of partial FSSP solutions and proposed a full list of the smallest four-state *symmetric* powers-of-2 FSSP protocols that can synchronize any one-dimensional (1D) ring cellular automata of length $n = 2^k$ for any positive integer $k \geq 1$. Afterwards, Ng [2011] also added a list of *asymmetric* FSSP partial solutions, thus completing the four-state powers-of-2 FSSP partial solutions. The number four is the lower bound in the class of FSSP protocols. A question: are there any other four-state partial solutions? remained. In this paper, we answer the question by proposing a new class of the smallest symmetric four-state FSSP protocols that can synchronize any 1D ring of length $n = 2^k - 1$ for any positive integer $k \geq 2$. We show that the class includes a rich variety of FSSP protocols that consists of 39 *symmetric* solutions, ranging from minimum-time to linear-time in synchronization steps. In addition, we make an investigation into several interesting properties of these partial solutions, such as swapping general states and a duality property between them.**

Keywords*—cellular automata; FSSP; synchronization.

I. INTRODUCTION

We study a synchronization problem that gives a finite-state protocol for synchronizing a large scale of cellular automata. The synchronization in cellular automata has been known as the Firing Squad Synchronization Problem (FSSP) since its development, in which it was originally proposed by J. Myhill in Moore [6] to synchronize some/all parts of self-reproducing cellular automata. The FSSP has been studied extensively for more than fifty years in [1]-[12].

The minimum-time (i.e., $(2n - 2)$-step) FSSP algorithm was developed first by Goto [4] for synchronizing any one-dimensional (1D) array of length $n \geq 2$. The algorithm needed many thousands of internal states for its realization. Afterwards, Waksman [11], Balzer [1], Gerken [3] and Mazoyer [5] also developed a minimum-time FSSP algorithm and reduced the number of states realizing the algorithm, each with 16, 8, 7 and 6 states. On the other hand, Balzer [1], Sanders [8] and Berthiaume et al. [2] have shown that there exists no four-state synchronization algorithm. Thus, an existence or non-existence of five-state FSSP protocol has been an open problem for a long time. Umeo, Kamikawa and Yunès [9] answered partially by introducing a concept of *partial versus full* FSSP solutions and proposing a full list of the smallest four-state symmetric powers-of-2 FSSP partial protocols that can synchronize any 1D ring cellular automata of length $n = 2^k$ for any positive integer $k \geq 1$. Afterwards, Ng [7] also added a list of asymmetric FSSP partial solutions, thus completing the four-state powers-of-2 FSSP partial solutions. A question: are there any other four-state partial solutions? remained.

In this paper, we answer the question by proposing a new class of the smallest four-state FSSP protocols that can synchronize any 1D ring of length $n = 2^k - 1$ for any positive integer $k \geq 2$. We show that the class includes a rich variety of FSSP protocols that consists of 39 symmetric solutions, ranging from minimum-time to linear-time in synchronization steps. In addition, we make an investigation into several interesting properties of these partial solutions, such as swapping general states and a duality between them.In Section 2, we give a description of the 1D FSSP on rings and review some basic results on ring FSSP algorithms. Section 3 presents a new class of the symmetric partial solutions for rings. Section 4 gives a summary and discussions of the paper.

II. FIRING SQUAD SYNCHRONIZATION PROBLEM ON RINGS

A. Definition of the FSSP on Rings

The FSSP on rings is formalized in terms of the model of cellular automata. Figure 1 shows a 1D ring cellular automaton consisting of n cells, denoted by C_i, where $1 \leq i \leq n$. All cells are identical finite state automata. The ring operates in lock-step mode such that the next state of each cell is determined by both its own present state and the present states of its right and left neighbors. All cells (*soldiers*), except one cell, are initially in the *quiescent* state at time $t = 0$ and have the property whereby the next state of a quiescent cell having quiescent neighbors is the quiescent state. At time $t = 0$ the cell C_1 (*general*) is in the *fire-when-ready* state, which is an initiation signal to the ring.

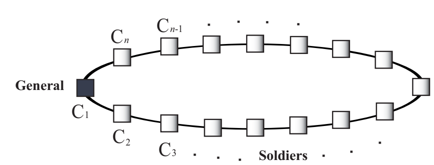


Fig. 1. One-dimensional (1D) ring cellular automaton

The FSSP is stated as follows: given a ring of n identical cellular automata, including a *general* cell which is activated at time $t = 0$, we want to give the description (state set and next-state transition function) of the automata so that, *at some future* *time*, all of the cells will *simultaneously* and, *for the first time*, enter a special *firing* state. The set of states and the next-state transition function must be independent of n. Without loss of generality, we assume $n \geq 2$. The tricky part of the problem is that the same kind of soldier having a fixed number of states must be synchronized, regardless of the length n of the ring.

A formal definition of the FSSP on ring is as follows: a cellular automaton $\mathcal{M}$ is a pair $\mathcal{M} = (\mathcal{Q}, \delta)$, where

1) $\mathcal{Q}$ is a finite set of states with three distinguished states $\mathtt{G}$, $\mathtt{Q}$, and $\mathtt{F}$. $\mathtt{G}$ is an initial general state, $\mathtt{Q}$ is a quiescent state, and $\mathtt{F}$ is a firing state, respectively.
2) δ is a next state function such that $\delta : \mathcal{Q}^3 \rightarrow \mathcal{Q}$.
3) The quiescent state $\mathtt{Q}$ must satisfy the following conditions: $\delta(\mathtt{Q}, \mathtt{Q}, \mathtt{Q}) = \mathtt{Q}$.

A ring cellular automaton $\mathcal{M}_n$ of length n, consisting of n copies of $\mathcal{M}$, is a 1D ring whose positions are numbered from 1 to n. Each $\mathcal{M}$ is referred to as a cell and denoted by C_i, where $1 \leq i \leq n$. We denote a state of C_i at time (step) t by $\mathtt{S}_i^t$, where $t \geq 0, 1 \leq i \leq n$. A *configuration* of $\mathcal{M}_n$ at time t is a function $\mathcal{C}^t : [1, n] \rightarrow \mathcal{Q}$ and denoted as $\mathtt{S}_1^t \mathtt{S}_2^t$ .... $\mathtt{S}_n^t$. A *computation* of $\mathcal{M}_n$ is a sequence of configurations of $\mathcal{M}_n$, $\mathcal{C}^0$, $\mathcal{C}^1$, $\mathcal{C}^2$, ..., $\mathcal{C}^t$, ..., where $\mathcal{C}^0$ is a given initial configuration. The configuration at time $t+1$, $\mathcal{C}^{t+1}$, is computed by synchronous applications of the next transition function δ to each cell of $\mathcal{M}_n$ in $\mathcal{C}^t$ such that:

$\mathtt{S}_1^{t+1} = \delta(\mathtt{S}_{n-1}^t, \mathtt{S}_1^t, \mathtt{S}_2^t)$, $\mathtt{S}_i^{t+1} = \delta(\mathtt{S}_{i-1}^t, \mathtt{S}_i^t, \mathtt{S}_{i+1}^t)$, for any i, $2 \leq i \leq n-1$, and $\mathtt{S}_n^{t+1} = \delta(\mathtt{S}_{n-1}^t, \mathtt{S}_n^t, \mathtt{S}_1^t)$.

A *synchronized configuration* of $\mathcal{M}_n$ at time t is a configuration $\mathcal{C}^t$, $\mathtt{S}_i^t = \mathtt{F}$, for any $1 \leq i \leq n$.

The FSSP is to obtain an $\mathcal{M}$ such that, for any $n \geq 2$,

1) A synchronized configuration at time $t = T(n)$, $\mathcal{C}^{T(n)} = \overbrace{\mathtt{F}, \cdots, \mathtt{F}}^{n}$ can be computed from an initial configuration $\mathcal{C}^0 = \mathtt{G}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-1}$.
2) For any t, i such that $1 \leq t \leq T(n) - 1$, $1 \leq i \leq n, \mathtt{S}_i^t \neq \mathtt{F}$.

B. Full vs. Partial Solutions

One has to note that any solution in the original FSSP problem is to synchronize any array of length $n \geq 2$. We call it **full** solution. Berthiaume et al. [2] presented an eight-state full solution for the ring. On the other hand, Umeo, Kamikawa, and Yunès [9] and Ng [7] constructed a rich variety of 4-state protocols that can synchronize some infinite set of rings, but not all. We call such protocol **partial** solution. Here, we summarize recent developments on small state solutions in the ring FSSP. Berthiaume, Bittner, Perkovic, Settle, and Simon [2] gave time and state lower bounds for the ring FSSP, described in Theorems 1, 2, and 3, below.

Theorem 1 (Time Lower Bound) The minimum time in which the ring FSSP could occur is no earlier than n steps for any ring of length n.

Theorem 2 There is no 3-state **full** solution to the ring FSSP.

Theorem 3 There is no 4-state, symmetric, minimal-time **full** solution to the ring FSSP.

Umeo, Kamikawa, and Yunès [9] introduced a class of partial solutions to the FSSP and showed that there exist 17 symmetric 4-state partial solutions to the ring FSSP.

Theorem 4 There exist **17 symmetric** 4-state partial solutions to the ring FSSP for the ring of length $n = 2^k$ for any positive integer $k \geq 1$.

Ng [7] added a list of 80 asymmetric 4-state solutions, this completing the powers-of-two solutions.

Theorem 5 There exist **80 asymmetric** 4-state partial solutions to the ring FSSP for the ring of length $n = 2^k$ for any positive integer $k \geq 1$.

C. A Quest for Four-State Partial Solutions for Rings

- **Four-state ring cellular automata**
  Let $\mathcal{M}$ be a four-state ring cellular automaton $\mathcal{M}$ = $\{\mathcal{Q}, \delta\}$, where $\mathcal{Q}$ is an internal state set $\mathcal{Q}$ = $\{\mathtt{A}, \mathtt{F}, \mathtt{G}, \mathtt{Q}\}$ and δ is a transition function such that $\delta : \mathcal{Q}^3 \rightarrow \mathcal{Q}$. Without loss of generality, we assume that $\mathtt{Q}$ is a quiescent state with a property $\delta(\mathtt{Q}, \mathtt{Q}, \mathtt{Q}) = \mathtt{Q}$, $\mathtt{G}$ is a general state, $\mathtt{A}$ is an auxiliary state and $\mathtt{F}$ is the *firing* state, respectively. The initial configuration is $\mathtt{G}\overbrace{\mathtt{QQ}, ..., \mathtt{Q}}^{n-1}$ for $n \geq 2$. We say that an FSSP solution is *symmetric* if its transition table has a property such that $\delta(x, y, z) = \delta(z, y, x)$, for any state x, y, z in $\mathcal{Q}$. Otherwise, the FSSP solution is called *asymmetric* one.

Q		Right State		
		Q	G	A
Left State	Q	Q	•	•
	G	•	•	•
	A	•	•	•

G		Right State		
		Q	G	A
Left State	Q	•	•	•
	G	•	•	•
	A	•	•	•

A		Right State		
		Q	G	A
Left State	Q	•	•	•
	G	•	•	•
	A	•	•	•

Fig. 2. Four-state transition table

- **A computer investigation into four-state FSSP solutions for rings**
  Figure 2 is a four-state transition table, where a symbol • shows a possible state in $\mathcal{Q} = \{\mathtt{A}, \mathtt{F}, \mathtt{G}, \mathtt{Q}\}$. Note that we have totally 4^{26} possible transition rules. We make a computer investigation into the transition rule set that might yield possible FSSP solutions. Our strategy is based on a backtracking searching. A similar technique was employed in Ng [7]. Due to the space available, we omit the details of the backtracking searching strategy. The outline of those solutions will be described in the next section.

III. Four-State Symmetric Partial Solutions

In this section, we will establish the following theorem with a help of computer investigation.

Theorem 6 There exist **39 symmetric** 4-state partial solutions to the ring FSSP for the ring of length $n = 2^k - 1$ for any positive integer $k \geq 2$.

 ISBN: 978-1-61208-608-8

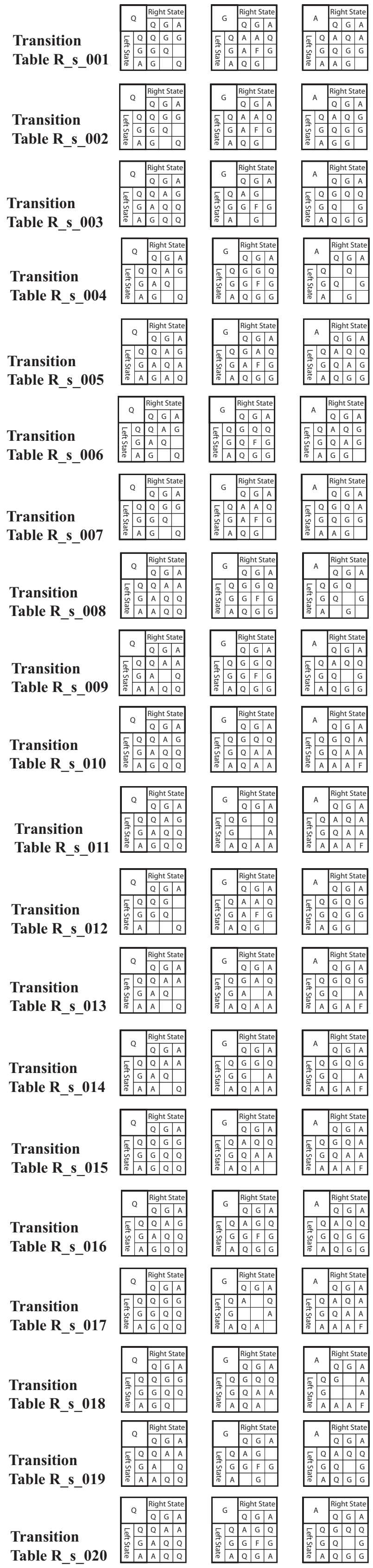


Fig. 3. Transition tables for 39 minimum-time, nearly minimum-time and non-minimum-time symmetric solutions

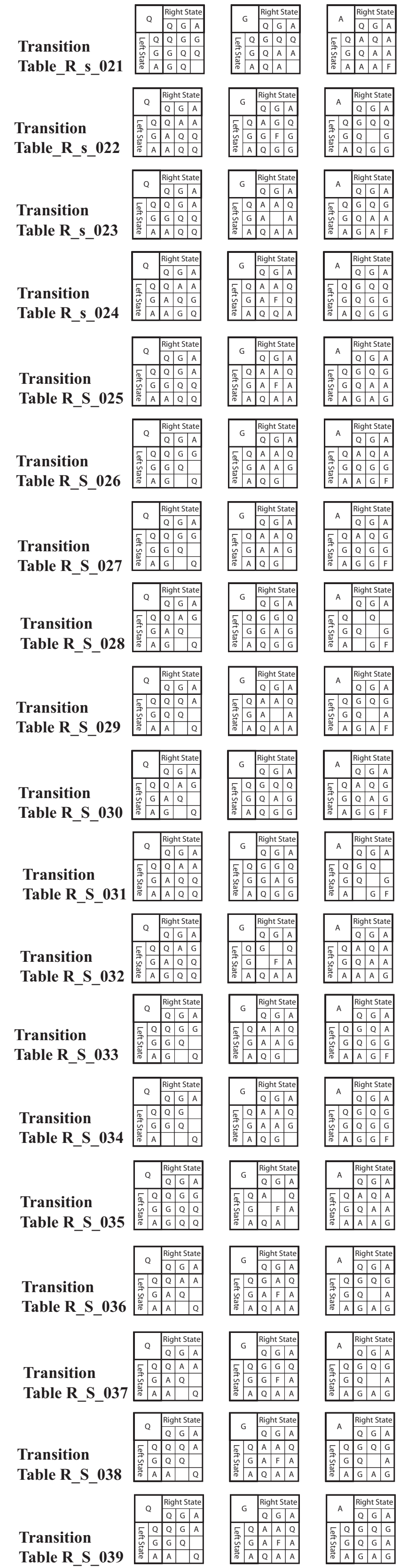


Fig. 4. Transition tables for 39 minimum-time, nearly minimum-time and non-minimum-time symmetric solutions

TABLE I. TIME COMPLEXITY AND NUMBER OF TRANSITION RULES FOR 39 SYMMETRIC PARTIAL SOLUTIONS

Symmetric Partial Solutions	Time Complexity	# of Transition Rules	Notes
$\mathrm{R}_{\mathrm{S\_1}}$	$T_G(n) = T_A(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_2}}$	$T_G(n) = T_A(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_3}}$	$T_G(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_4}}$	$T_G(n) = n$	20	
$\mathrm{R}_{\mathrm{S\_5}}$	$T_G(n) = n$	27	
$\mathrm{R}_{\mathrm{S\_6}}$	$T_G(n) = n$	24	
$\mathrm{R}_{\mathrm{S\_7}}$	$T_G(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_8}}$	$T_G(n) = T_A(n) = n$	24	
$\mathrm{R}_{\mathrm{S\_9}}$	$T_G(n) = T_A(n) = n$	25	
$\mathrm{R}_{\mathrm{S\_10}}$	$T_G(n) = T_A(n) = n$	27	
$\mathrm{R}_{\mathrm{S\_11}}$	$T_G(n) = T_A(n) = n$	24	
$\mathrm{R}_{\mathrm{S\_12}}$	$T_G(n) = n$	21	
$\mathrm{R}_{\mathrm{S\_13}}$	$T_G(n) = T_A(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_14}}$	$T_G(n) = T_A(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_15}}$	$T_G(n) = T_A(n) = n$	26	
$\mathrm{R}_{\mathrm{S\_16}}$	$T_G(n) = T_A(n) = n$	27	
$\mathrm{R}_{\mathrm{S\_17}}$	$T_G(n) = T_A(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_18}}$	$T_G(n) = T_A(n) = n$	22	
$\mathrm{R}_{\mathrm{S\_19}}$	$T_G(n) = T_A(n) = n$	22	
$\mathrm{R}_{\mathrm{S\_20}}$	$T_G(n) = n$	26	
$\mathrm{R}_{\mathrm{S\_21}}$	$T_G(n) = T_A(n) = n$	25	
$\mathrm{R}_{\mathrm{S\_22}}$	$T_G(n) = T_A(n) = n$	26	
$\mathrm{R}_{\mathrm{S\_23}}$	$T_G(n) = T_A(n) = n$	26	
$\mathrm{R}_{\mathrm{S\_24}}$	$T_G(n) = n$	27	
$\mathrm{R}_{\mathrm{S\_25}}$	$T_G(n) = T_A(n) = n+1$	27	
$\mathrm{R}_{\mathrm{S\_26}}$	$T_G(n) = T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_27}}$	$T_G(n) = T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_28}}$	$T_G(n) = n+1$	22	
$\mathrm{R}_{\mathrm{S\_29}}$	$T_G(n) = n+1, T_A(n) = n$	23	
$\mathrm{R}_{\mathrm{S\_30}}$	$T_G(n) = n+1$	25	
$\mathrm{R}_{\mathrm{S\_31}}$	$T_G(n) = T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_32}}$	$T_G(n) = T_A(n) = n+1$	25	
$\mathrm{R}_{\mathrm{S\_33}}$	$T_G(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_34}}$	$T_G(n) = n+1$	22	
$\mathrm{R}_{\mathrm{S\_35}}$	$T_G(n) = T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_36}}$	$T_G(n) = T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_37}}$	$T_G(n) = T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_38}}$	$T_G(n) = n+2, T_A(n) = n+1$	24	
$\mathrm{R}_{\mathrm{S\_39}}$	$T_G(n) = (3n+1)/2, T_A(n) = n+1$	25	

Let $\mathrm{R}_{\mathrm{S\_}i}, 1 \leq i \leq 39$ be a transition table for symmetric solutions obtained in this paper. We refer to the ith symmetric transition table as symmetric solution i, where $1 \leq i \leq 39$. The details are as follows:

- **Symmetric Minimum-Time Solutions:**
  We have got 24 minimum-time symmetric partial solutions operating in exactly $T(n) = n$ steps. We show their transition rules $\mathrm{R}_{\mathrm{S\_}i}, 1 \leq i \leq 24$ in Figures 3 and 4.
- **Symmetric Nearly Minimum-Time Solutions:**
  We have got 14 nearly minimum-time symmetric partial solutions operating in $T(n) = n + O(1)$ steps. Their transition rules $\mathrm{R}_{\mathrm{S\_}i}, 25 \leq i \leq 38$ are given in Figure 4. Most of the solutions, that is, solutions 25-37 operate in $T(n) = n + 1$ steps. The solution 38 operates in $T(n) = n + 2$ steps.
- **Symmetric Non-Minimum-Time Solution:**
  It is seen that one non-minimum-time symmetric partial solution 39 exists. Its time complexity is $T(n) = (3n + 1)/2$. The transition rule $\mathrm{R}_{\mathrm{S\_39}}$ is given in Fig. 4.

In Table I, we give the time complexity and number of transition rules for each symmetric solution.

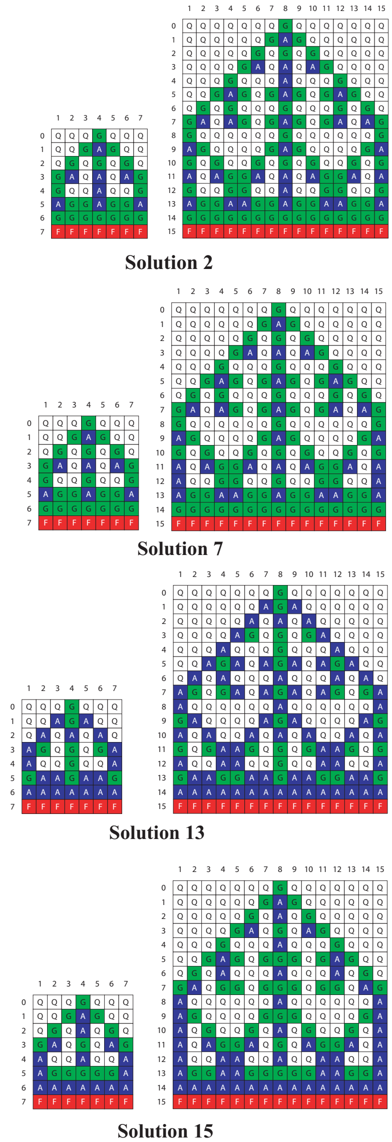


Fig. 5. Snapshots on 7 and 15 cells for symmetric solutions 2, 7, 13, and 15

Here, we give some snapshots on 7 and 15 cells for minimum-time, nearly minimum-time and non-minimum-time FSSP solutions, respectively, in Figures 5, 6, and 7.

Now, we give several interesting observations obtained for the rule set.

Observation 1 (Swapping General States)
It is noted that some solutions have a property that both of the states G and A can be an initial general state without introducing any additional transition rules and yield successful synchronizations from each general state.

For example, solution 1 can synchronize any ring of length $n = 2^k - 1, k \geq 2$ in $T(n) = n$ steps from both an initial configuration $\mathtt{G}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-1}$ and $\mathtt{A}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-1}$, respectively. Let $T_{G-R_{S\_i}}(n)$ (or simply $T_G(n)$, if the rule number is specified) and $T_{A-R_{S\_i}}(n)$ ($T_A(n)$) be synchronization steps staring the solution $\mathrm{R}_{\mathrm{S\_}i}$ from the state G and A, respectively, for rings

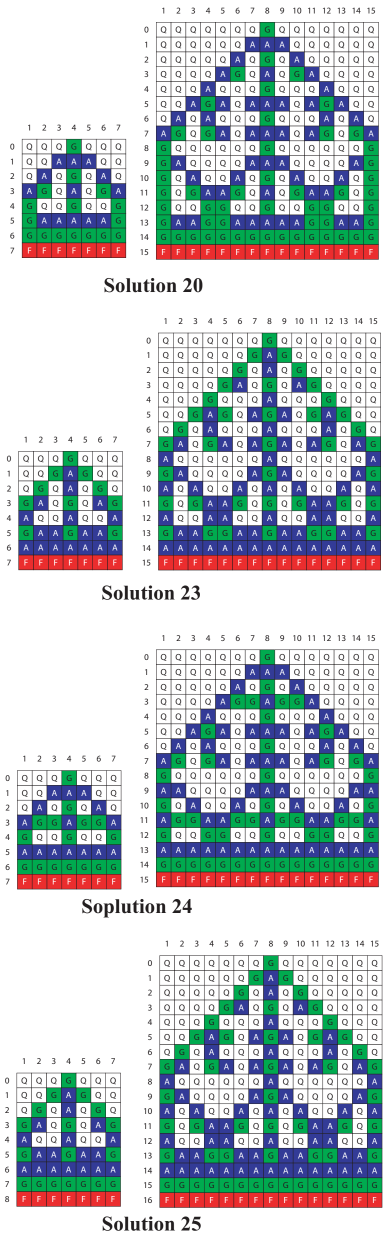


Fig. 6. Snapshots on 7 and 15 cells for symmetric solutions 20, 23, 24, and 25

of length n. Then, we have $T_{G-R_{S\_1}}(n) = T_{A-R_{S\_1}}(n) = n$.

In Fig. 8, we show some synchronized configurations on 3, 7, and 15 cells with a general G (left) and A (right), respectively, for the solution 1. The observation doesn't always hold for all symmetric rules. For example, the solution 3 can synchronize any ring of length $n = 2^k - 1, k \geq 2$ in $T(n) = n$ steps from the general state G, but not from the state A.

The Observation 1 yields the following *duality* relation among the four-state rule sets.

Observation 2 (Duality)
Let x and y be any four-state FSSP solution for rings and x is obtained from y by swapping the states G and A in y and vice versa. We say that the two rules x and y are dual concerning the states G and A. The relation is denoted as $x \leftrightarrows y$. We have:

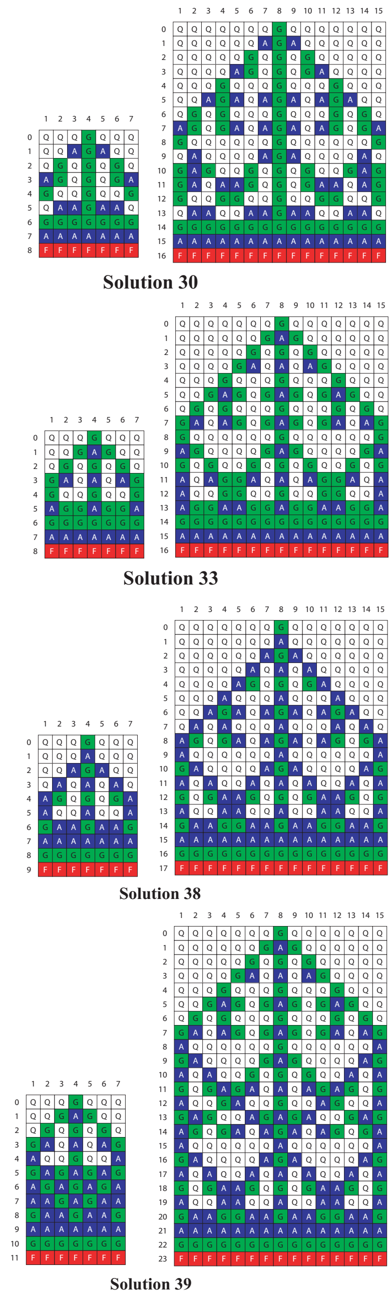
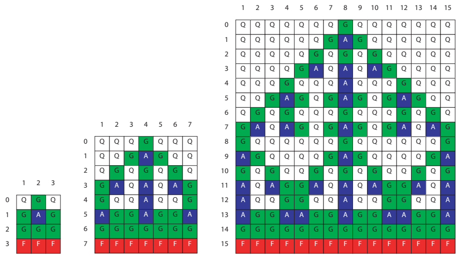


Fig. 7. Snapshots on 7 and 15 cells for symmetric solutions 30, 33, 38, and 39

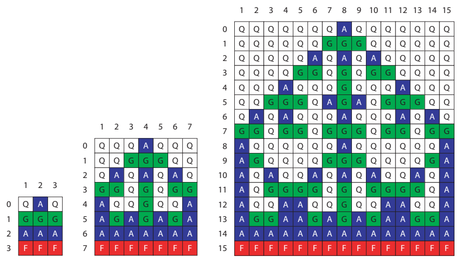
$$\begin{array}{c} R_{S\_1} \leftrightarrows R_{S\_14},\ R_{S\_2} \leftrightarrows R_{S\_13}, \\ R_{S\_8} \leftrightarrows R_{S\_17},\ R_{S\_9} \leftrightarrows R_{S\_21}, \\ R_{S\_10} \leftrightarrows R_{S\_16},\ R_{S\_15} \leftrightarrows R_{S\_22}, \\ R_{S\_18} \leftrightarrows R_{S\_19},\ R_{S\_26} \leftrightarrows R_{S\_37}, \\ R_{S\_27} \leftrightarrows R_{S\_36},\ R_{S\_31} \leftrightarrows R_{S\_35}. \end{array}$$

IV. Summary and Discussions

A quest for smaller state FSSP solutions has been an interesting problem for a long time. We have answered to the



Solution 1 with a general-state G



Solution 1 with a general-state A

Fig. 8. Synchronized configurations on 3, 7, and 15 cells with a general-state G (upper) and A (lower), respectively, for the Solution 1

question by proposing a new class of the smallest four-state FSSP protocols that can synchronize any 1D ring of length $n = 2^k - 1$ for any positive integer $k \geq 2$. We show that the class includes a rich variety of FSSP protocols that consists of 39 symmetric solutions, ranging from minimum-time to linear-time in synchronization steps. Some interesting properties in the structure of 4-state partial solutions have been discussed. We strongly believe that no smallest solutions exist other than the ones proposed for length 2^k rings in Umeo, Kamikawa and Yunès [9] and Ng [7] and for rings of length $2^k - 1$ in this paper. A question: how many 4-state partial solutions exist for arrays (open-rings)? remains open. We think that there would be a large number of the smallest 4-state partial solutions for arrays. Its number would be larger than several thousands. The structure of the 4-state array partial synchronizers is far more complex than the 4-state ring partial synchronizers.

References

[1] Balzer, R.: An 8-state minimal time solution to the firing squad synchronization problem. *Information and Control*, **10**, 22-42 (1967)

[2] Berthiaume, A., Bittner, T., Perković, L., Settle, A., and Simon, J.: Bounding the firing synchronization problem on a ring. *Theoretical Computer Science*, 320, pp. 213-228 (2004)

[3] Gerken, H. D.: Über Synchronisationsprobleme bei Zellularautomaten. *Diplomarbeit, Institut für Theoretische Informatik, Technische Universität Braunschweig*, pp. 1-50 (1987)

[4] Goto, E.: A minimal time solution of the firing squad problem. *Dittoed course notes for Applied Mathematics* 298 (with an illustration in color), Harvard University, pp. 52-59 (1962)

[5] Mazoyer, J.: A six-state minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, **50**, 183-238 (1987)

[6] Moore, E. F.: The firing squad synchronization problem. In *Sequential Machines, Selected Papers* (E. F. Moore, ed.), Addison-Wesley, Reading MA., pp. 213-214 (1964)

[7] Ng, W. L.: Partial solutions for the firing squad synchronization problem on rings. ProQuest publications, Ann Arbor, MI, pp.1-363, (2011)

[8] Sanders, P.: Massively parallel search for transition-tables of polyautomata. In *Proc. of the VI International Workshop on Parallel Processing by Cellular Automata and Arrays*, (C. Jesshope, V. Jossifov and W. Wilhelmi (editors)), Akademie, pp. 99-108 (1994)

[9] Umeo, H., Kamikawa, N., and Yunès, J.-B.; A family of smallest symmetrical four-state firing squad synchronization protocols for ring arrays. *Parallel Processing Letters*, Vol.19, No. 2. pp. 299-313 (2009)

[10] Umeo, H. and Yanagihara, T.: A smallest five-state solution to the firing squad synchronization problem. *Proc. of 5th Intern. Conf. on Machines, Computations, and Universality, MCU 2007*, LNCS 4664, pp. 292-302 (2007)

[11] Waksman, A.: An optimum solution to the firing squad synchronization problem. *Information and Control*, **9**, pp. 66-78 (1966)

[12] Yunès, J. B.: A 4-states algebraic solution to linear cellular automata synchronization. *Information Processing Letters*, Vol. 19, Issue 2, pp. 71-75 (2008)

Energy Efficient Real-time Scheduling Algorithm for Microcontroller Units

Douglas Lautner, Xiayu Hua, Miao Song, Scott DeBates, Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
e-mail:{dlautner, xhua}@hawk.iit.edu, {msong2, scottdebates}@motorola.com, ren@iit.edu

***Abstract*—Mobile smart devices are advancing with stronger demands of high energy efficiency and longer battery life. Utilizing energy-efficient microcontroller units (MCU) in a mobile device for always-on functionalities is proved to be an effective solution. MCUs have the ability to switch between different running modes dynamically enabling them to have outstanding low power performance while performing real-time sensing tasks. Besides hardware optimization, balancing energy efficiency and quality of service on a MCU lies within a well designed scheduling algorithm. In this paper, we formally define, model and derive a proper scheduling algorithm that guarantees that the task set is schedulable and minimizes power consumption. Our findings open up additional research of optimizing real-time scheduling algorithms with less energy consumption on an MCU while guaranteeing the quality of service, i.e., the schedulability of the given real-time task set.**

Keywords–Mobile device; embedded system; energy efficiency; real-time scheduling

I. INTRODUCTION

Battery life is one of the most important features for mobile smart devices. To improve the energy efficiency, mobile smart devices introduced low-power processors, such as microcontroller unit (MCU) based sensor fusion core [1], to perform non-critical control and calculation tasks with lower power consumption due to its special design. Placing a MCU into ”shutdown” mode, i.e., turn off the MCU, can stop the power consumption. However, shutting down and rebooting the MCU not only introduces extra energy consumption, but also causes execution delay that may affect the tasks' real-time requirements. On the other hand, keeping the MCU in the standby and idle mode continuously consumes energy, however it guarantees tasks can be executed without delay. schedule the shutdown and reboot on a MCU with the goal of 1) guaranteeing the schedulability of hard real-time tasks and 2) minimizing the energy consumption is a big challenge.

In terms of saving energy by changing the processor into different modes, Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are two popular research areas [2]–[4]. Most existing DVFS-related work focuses on minimizing the energy consumption while optimizing the quality of service [5]–[7]. Also, assumptions such that resources are always available during performance change [8], or the performance can be switched immediately to any level between the low and maximal performance [9] make existing real-time scheduling analyses with DVFS-enabled resource compliment to our case. While DVFS requires the resource to be always available, DPM-based solutions allow the system to shutdown the resource [3]. DPM solutions can also be implemented with DVFS (if the hardware allows) [3] to further reduce the energy consumption. However, most existing work on DPM targets on minimizing the task set's makespan or maximizing the system's throughput while optimizing the energy consumption. To the best of our knowledge, scheduling hard real-time tasks on a MCU and maximizing the MCU's energy efficiency is still a challenge yet to be solved.

Inspired by the existing work on DVFS and DPM scheduling issues, we investigate, discuss and analyze the challenges of deploying hard real-time tasks on MCU in an energy efficient way, given MCU's ability to switching running mode between standby and shutdown. In particular, we provide the energy-saving condition under which shutting down the MCU saves more energy than leaving it in standby mode and idle. Without creating an impact to the performance, we require the shutdowns should not break the schedulability of the given task set. We have further studied three possible approaches to specifically schedule shutdown time and duration for a MCU to guarantee the task set's schedulability while minimizing the overall energy consumption.

The rest of paper is organized as follows. In Section II, we propose the task model, the resource assumptions and the research goal. Section III proposes and analyzes our three possible solutions in detail. In Section IV, we conclude our current work and propose future extensions.

II. MODELING AND PROBLEM FORMULATION

In this section, we define the task and resource models, respectively, and provide the formal problem formulation.

A. Task Model

We use the task model defined in [10] which makes the following assumptions: 1. Tasks are periodically available with a fixed period; 2. Each task requires a fixed amount of time to finish; 3. Each task's relative deadline (with respect to the time that it becomes available) is the task's period. Specifically,

 ISBN: 978-1-61208-608-8

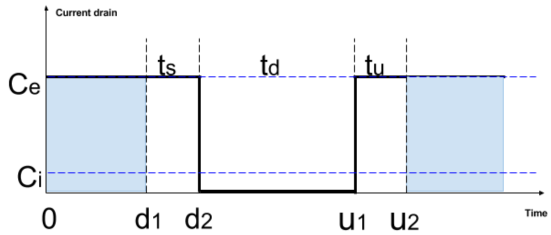


Figure 1. Shutdown time requirement

a task is defined as $\tau(e, p)$ where e is the longest possible execution time, p is the period.

B. MCU Resource Assumptions

For a MCU resource, it may have several running modes. In this work, we assume that a MCU resource has two modes: *standby* mode that can execute jobs immediately, but consumes power even when idle; *shutdown* mode that cannot execute jobs and does not consume power.

We further assume that a switch between the two modes has time overhead and consumes extra energy.

As the resource model varies in different possible solutions, we define the resource models in each solution individually.

C. Problem Formulation

For a given task set and a MCU resource, schedule the switch of resource modes between standby mode and shutdown mode to: 1) Guarantee that each job of a task in the task set meets its deadline, and 2) Minimize the power consumption.

III. Possible Solutions

In this section, we first calculate the minimal shutdown duration requirement such that a shutdown behavior: 1) Is practical for a MCU to perform from a hardware perspective and 2) Does save power compared to leaving the MCU idle. Then, we provide three possible solutions, as well as our preliminary research results, of how to schedule the shutdown for a MCU with running real-time task sets. For the first solution, we have finished the theoretical analysis for tasks under Earliest Deadline First scheduler. For the Rate Monotonic scheduler case, as well as the remaining two possible solutions, they are work in progress and we will research them in our future work.

A. The minimal shutdown duration requirement

As illustrated in Figure 1, to perform a shutting down, the MCU stops providing computing services to the tasks first (at time d_1). Before cutting the power (at time point d_2), the MCU needs to backup its runtime status, i.e, values in registers or data in the RAM, into storage in state t_s. During time interval t_d, the MCU is in shutdown mode. At time point u_1, the MCU starts the rebooting procedure during t_u and is completed at u_2. In the following sections, we call the MCU as a *computing resource* and formally define the resource model of MCU in the following definition.

Definition 1. *A MCU resource R is defined as a quad notated as $R(t_s, t_u, C_i, C_e)$ where t_s is the time duration of switching from running mode to shutdown mode and t_u is the time duration of switching from shutdown mode to running mode. C_i and C_e are the current drains when the MCU is idle or is calculating, respectively.*

As the MCU's shutdown and reboot procedure needs time and extra energy, we derive the minimal shutdown time in the following lemma.

Lemma 1. *For a given MCU resource $R(t_s, t_u, C_i, C_e)$, the minimal shutdown time T_s that saves energy is determined as:*

$$T_s > \frac{(t_s + t_u) \cdot C_e}{C_i} \tag{1}$$

Proof. As Figure 1 illustrates, the time interval that the MCU is unavailable to execute tasks starts at time point d_1 and ends at u_2, i.e., $T_s = u_2 - d_1$. If the resource is in idle time, it consumes $(u_2 - d_1) \cdot C_i$ power during this time period. On the other hand, if the MCU is shut down, during t_s it consumes $t_s \cdot C_e$ power and during t_u it consumes $t_u \cdot C_e$ power. Compared to leaving the MCU idle, shutting down the MCU saves more energy when the following condition is satisfied.

$$(u_2 - d_1) \cdot C_i > (t_s + t_u) \cdot C_e$$

Hence

$$T_s > \frac{(t_s + t_u) \cdot C_e}{C_i} \tag{2}$$

Obviously, T_s should also be longer than $t_s + t_u$ to have enough time to shutdown and then reboot the MCU. Since $C_e > C_i$, (2) also implies $T_s > t_s + t_u$. □

B. Handle the System Shutdown: When the Mode Switch is periodic

The first approach is to shutdown the MCU periodically. As illustrated in Figure 3, a periodic shutdown is performed with the period and duration of 5 and 2 time units, respectively, i.e., $S(2, 5)$. As a comparison, the case of scheduling on regular resource is illustrated in Figure 2. The advantage of this approach is obvious. To implement this approach, the scheduler only needs a timer, therefore this approach consumes minimal CPU resource compared to the approach we will study in the next section. The major challenge is to determine the shutdown duration time and the shutdown period that guarantees the schedulability of the given task set while minimizes the overall power consumption.

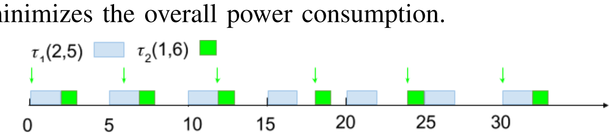


Figure 2. Regular resource

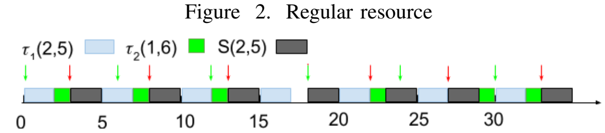


Figure 3. Periodic resource

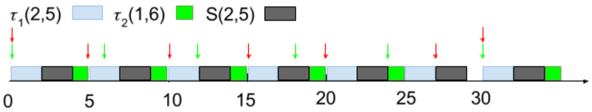


Figure 4. Resource with shutdown task insertion

Under this approach, the resource's properties match the definition of a periodic resource [11]–[13], with respect to the task set, as it switches periodically between available and unavailable. More specifically, this resource is categorized to a *fixed-pattern periodic resource* in [12] and [13]. Compared to the existing scheduling research work on periodic resources, the challenges of our current problem is to determine both the shutdown frequency and the shutdown duration which are assumed to be given in the existing work.

In our work, we calculate these two parameters to guarantee the task set's schedulability (if schedulable on a resource without shutting down) while minimizing the energy consumption. Specifically, we leverage the schedulability analyses in [12] and [13], change the resource period and downtime into variables, determine their solution domains with respect to the task set's schedulability and then find the optimal (or near optimal) solution in the solution domain that minimizes the power consumption.

A possible approach to find the optimal solution is to derive the schedulability condition from periodic resource bounds [11]. From the necessary schedulability bound, we can derive the dependency between the shutdown period and its duration under the schedulability requirement. Then, based on this dependency, we further use heuristic ways (to deliver a near optimal solution) to search the optimal pair of period and duration values.

To perform the theoretical analysis, the pessimistic way is to first calculate the bound, and use the bound to further calculate the θ parameters. Following Lemmas and theorems provide the shutdown time bound for the EDF scheduler. For convenience, we reuse the MCU notation R and redefine the resource model in the following definition for schedulability analysis pruposes.

Definition 2. *A MCU resource with periodic sleep is defined as a tuple and is denoted as $R(\theta, \pi)$ where θ is the resource available time, with respect to the task set, and π is the system sleep period.*

Note that both t_s and t_u time duration is not counted in θ as tasks are not able to be executed during these two durations.

Lemma 2. *For a given resource $R(\theta, \pi)$ and a task set $T = \{\tau_1, ..., \tau_n\}$ where τ_i represents a independent task $\tau_i(e_i, p_i)$. The minimal task period in T is denoted as p_{min}. With Earliest Deadline First scheduler, T is guaranteed to be schedulable if the following relation is valid: $\pi < \frac{p_{min} \cdot (\theta - U_T) + \theta^2}{\theta}$, where U_T is the total utilization rate of T, i.e., $U_T = \sum_{i-1}^{n} \frac{e_i}{p_i}$.*

Proof. We first introduce the linear demand bound function $ldbf(t) = U_T \cdot t$ for EDF from [14], [15]. This function describes that for any time interval with given length t, the overall resource demand, such like the overall CPU cycles needed, will not acceed $ldbf(t)$.

We then calculate in the worst case, at least how much resource can be offered by the MCU with periodic shutdown in a given time length t, which is calculated as the supply bound function $sbf(t)$. Obviously, for any time interval with same length, the worst case is that this time interval starts with a shutting down, as illustrated in Figure 5.

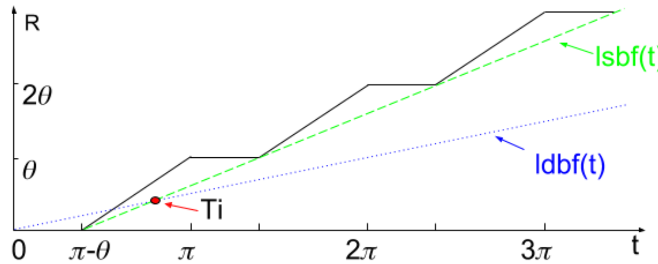


Figure 5. Demand and supply bound functions

Specifically, from time 0 to time θ, the job receives no resource. At time θ to the end of this resource period, the MCU is up and running and executes the job. As the MCU periodically turns on and off, this procedure repeats in each MCU period. We formalize this periodic procedure as:

$$sbf(t) = \begin{cases} 0, if\ t < \pi - \theta, \\ \lfloor \frac{t}{\pi} \rfloor \cdot \theta + max(0, (t\ mod\ \pi - (\pi - \theta))), otherwise \end{cases} \quad (3)$$

In (3), if the time interval is shorter than $\pi - \theta$, then in the worst case, the MCU is shut down during the entire time interval and hence there is zero resource provided by the MCU in this time interval. In the case that $t \geq \pi - \theta$, we first calculate how many complete MCU peirods are in this interval, i.e., $\lfloor \frac{t}{\pi} \rfloor$. Since for one complete MCU period, MCU provides θ executable time, this parts contributes $\lfloor \frac{t}{\pi} \rfloor \cdot \theta$ executable time. For the remaining part, i.e., $t\ mod\ \pi$, it is shorter than a period. If it's also shorter than the shutdown time, in the worst case this part contributes no executable time, otherwise it provides $t\ mod\ (\pi - \theta)$ executable time.

Since (3) is a step-function which is difficult to conduct further proof, we further derivate the linear supply bound function $lsbf(t) = \frac{\theta}{\pi} \cdot t - \theta + \frac{\theta^2}{\pi}$.

The green dot line in Figure 5 illustrates the intuition of $lsbf(t)$.

To guarantee the schedulability of a task set, the following condition should be satisfied [15], [16], i.e., when the length of a time interval is equal or longer than p_{min}, then $lsbf(t) \geq ldbf(t)$. With this condition, we derive the final conclusion. First, with (3), $ldbf(t)$ and $lsbf(t) \geq ldbf(t)$ we have

$$\frac{\theta}{\pi} \cdot t - \theta + \frac{\theta^2}{\pi} > U_T \cdot t \quad \rightarrow \quad t \geq \frac{\pi\theta - \theta^2}{\theta - U_T}$$

As illustrated in Figure 5, we denote the time point $\frac{\pi\theta - \theta^2}{\theta - U}$ as T_i. The intuition of T_i is that if a time interval is longer than T_i, then the minimal resource supply is guaranteed to be larger than the maximal possible resource demand. To guarantee the schedulability of the task set, we only need to guarantee that for a time interval with length p_{min}, the resource supply is no less than the demand, therefore we have:

$$t \geq \frac{\pi\theta - \theta^2}{\theta - U} \rightarrow p_{min} \geq \frac{\pi\theta - \theta^2}{\theta - U}$$
$$\rightarrow \pi < \frac{p_{min} \cdot (\theta - U_T) + \theta^2}{\theta} \quad (4)$$

□

With Lemma 1, we now have the relation between π and θ.

Theorem 1. *For a given resource $R(\theta, \pi)$ and a task set $T = \{\tau_1, ..., \tau_n\}$ where τ_i represents a independent task $\tau_i(e_i, p_i)$. The minimal task period in T is denoted as p_{min}. With Earliest Deadline First scheduler, T is guaranteed to be schedulable if the following relation is valid:*

$$\begin{cases} \pi < \frac{p_{min} \cdot (\theta - U_T) + \theta^2}{\theta} \\ \theta > \frac{(t_s + t_u) \cdot C_m}{C_i} \end{cases} \quad (5)$$

where U_T is the total utilization rate of T, i.e., $u = \sum_{i-1}^{n} \frac{e_i}{p_i}$.

With Theorem 1, we create the solution domain for π and θ. If there is no solution in this domain, then the task set is non-schedulable even without a MCU shutdown. If this domain is non-empty, the next step is to find the optimal solution ,i.e. (π, θ) value pair that can minimize the power consumption in a T's hyperperiod. The approach could be either theoretical or heuristic searching approaches, we will further study in our future work.

C. Handle the System Shutdown: Treat the Shutdown as a Special Task

Aside from turning down the MCU periodically, another possible solution is to treat the turning down as a special task. As illustrated in Figure 4, a periodic shutdown $S(2, 5)$, i.e., shutdown two time units for every five time units, is treated as a regular task. Specifically, we first determine the shutdown duration e and shutdown period p and insert the shutdown into the original task set as a special task. We then use the original scheduler to schedule both the new task set with the special task. This solution is applicable for the classical schedulers, such as RM or EDF, without changing the original system too much. Also, comparing to the periodic shutdown, the shutdown is more flexible, i.e., the shutdown can be arranged to strart from anytime within its perioid, therefore it should have both higher system utilization rate and energy efficiency compared to the periodic shutdown approach.

This solution has two challenges. First, the shutdown task is non-preemptive. Second, we still need to determine the e' and p' of the shutdown task to 1) guarantee the task set's schedulability and 2) minimize p' while maximizing e'.

For the first challenge, one possible approach is to align the shutdown task's period p' to a certain task and extends the chosen task's execution time. For example, when using a Rate Monotonic scheduler, if we align the shutdown task to the task with shortest period (i.e., highest priority) $\tau_1(e_1, p_1)$, then we have a new τ_1' where $e_1' = e_1 + e'$. After this transition, the number of tasks remains unchanged and the shutdown task is non-preemptive as it is always accompanied by the execution of τ_1 which is non-preemptive to all other tasks. Then, the problem changes to determine the longest e_i' that can still guarantee the task set's schedulability.

This approach may guarantee the schedulability and handle the non-preemptive property of the shutdown task, however the second challenge is still unsolved.

D. Handle the System Shutdown: New Scheduling Algorithm

The third possible approach is to design a new algorithm which can dynamically switch the MCU into sleep mode. This approach has the highest effect with respect to the energy efficiency as it can utilize all possible chances to put the MCU into sleep mode. However, it also has the most difficult challenges.

For this approach, the immediate problem is for the scheduler to know whether an incoming idle time is long enough to switch the MCU into sleep mode and then switch back before the release of the next job. As current schedulers only handle resource conflicts, i.e, when more jobs are ready to run but only less resources are available, a scheduler determines which jobs to run. Under this context, a scheduler is not able to be aware of the incoming job running situations. Therefore, new improvements are required for the scheduling algorithms to further determine the MCU's running modes.

The implementation of this approach is most likely to be heuristic as it is essentially an integrated linear programming solution and it is expensive resource wise. The basic approach of our solution could be to find the optimal schedule first using an integrated linear solution, and then gradually release the conditions to reach a near optimal solution, which balances both the scheduling overhead and the energy efficiency.

IV. Conclusion and Future Work

In this work, we formally formulated the scheduling problems of deploying periodic hard real-time tasks on a MCU that has multiple running modes. To achieve the goal of 1) Guaranteeing the task set's schedulability and 2) Maximizing the energy efficiency, we further provided analyses of three possible solutions, i.e., 1) Periodic shutdown, 2) Treat shutdown as a special task, and 3) Design a new scheduling algorithm. In our future work, we will continually study each of these three possible solutions and find the optimal ones that save the maximal energy while guaranteeing the schedulability.

References

[1] D. Lautner, X. Hua, S. Debates, M. Song, J. Shah, and S. Ren, "Baas (bluetooth-as-a-sensor): conception, design and implementation on mobile platforms," in *Proceedings of the Symposium on Applied Computing*. ACM, 2017, pp. 550–556.

[2] R. David, P. Bogdan, and R. Marculescu, "Dynamic power management for multicores: Case study using the intel scc," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*. IEEE, 2012, pp. 147–152.

[3] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, p. 111, 2014.

[4] S. Mittal, "Power management techniques for data centers: A survey," *arXiv preprint arXiv:1404.6681*, 2014.

[5] J. R. Lorch and A. J. Smith, “Improving dynamic voltage scaling algorithms with pace,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[6] C. Xian, Y.-H. Lu, and Z. Li, “Dynamic voltage scaling for multitasking real-time systems with uncertain execution time,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1467–1478, 2008.

[7] J.-J. Chen, N. Stoimenov, and L. Thiele, “Feasibility analysis of on-line dvs algorithms for scheduling arbitrary event streams,” in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*. IEEE, 2009, pp. 261–270.

[8] K. Choi, R. Soma, and M. Pedram, “Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times,” *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 24, no. 1, pp. 18–28, 2005.

[9] R. Jejurikar and R. Gupta, “Energy aware task scheduling with task synchronization for embedded real time systems,” in *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*. ACM, 2002, pp. 164–169.

[10] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[11] I. Shin and I. Lee, “Compositional real-time scheduling framework with periodic model,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, p. 30, 2008.

[12] A. K. Mok, X. Feng, and D. Chen, “Resource partition for real-time systems,” in *Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE*. IEEE, 2001, pp. 75–84.

[13] S. Shirero, M. Takashi, and H. Kei, “On the schedulability conditions on partial time slots,” in *Real-Time Computing Systems and Applications, 1999. RTCSA’99. Sixth International Conference on*. IEEE, 1999, pp. 166–173.

[14] A. Easwaran, M. Anand, and I. Lee, “Compositional analysis framework using edp resource models,” in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*. IEEE, 2007, pp. 129–138.

[15] X. Hua, C. Guo, H. Wu, D. Lautner, and S. Ren, “Schedulability analysis for real-time task set on resource with performance degradation and periodic rejuvenation,” in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on*. IEEE, 2015, pp. 197–206.

[16] X. Feng, “Design of real-time virtual resource architecture for large-scale embedded systems,” Ph.D. dissertation, 2004.

Representing Medication Guidelines for Use in Production Rule Systems in the Context of Polypharmacy

Jonas W. Könning*, Carlos A Velasco† and Yehya Mohamad†, Stefan Decker‡ and Oya Beyan‡
*RWTH Aachen University, Aachen, Germany
Email: jonas.koenning@rwth-aachen.de
†Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany
Email: {carlos.velasco},{yehya.mohamad}@fit.fraunhofer.de
‡RWTH Aachen University, Aachen, Germany
Email: {beyan},{decker}@dbis.rwth-aachen.de

***Abstract*—This paper presents an approach to represent medication guidelines in a machine readable form for its use within a home care environment for elderly patients with chronic diseases. The overall system comprises a patient-centred integrated care environment, supported by computerized systems to improve the quality of home hospitalization. One component of the system is a decision support system for improving the medication of elderly patients. For this purpose, a machine actionable version of standardized medication guidelines was required. However, such version was not available, the translation of the guidelines from human readable into machine readable rules and the implementation are complex and pose many challenges e.g. ambiguity of the rules. Scope of this work included the design and implementation of such a rule base. Guidelines were selected based on their prominence in the domain and analyzed for their structure to allow for the creation of templates, which can be used in the automatic generation of rules. The templates were designed to work with the Drools business rule management system. While still simple, the current prototype shows good performance and potential for future extensions.**

***Index Terms*—health information management, rule-based decision-making system, polypharmacy, drug delivery, drug-drug interactions, comorbidity**

I. Introduction

The problems originated by an aging population have deserved greater attention by a multitude of organizations, notably in the World Health Organization (WHO) report "Global Health and Aging" in 2011 [1]. The number of people aged 65 or older is projected to reach 1.5 billion in 2050, from 524 million reported in 2010. Together with the eradication of infectious diseases and parasites in most parts of the world, which were especially dangerous for infants and children, chronic noncommunicable diseases are now the biggest burden on health and on the health care systems. Elder patients tend to suffer from so called co-morbid illnesses (or *comorbidity*), i.e., the presence of additional diseases or disorders that exist concurrently with a primary disease. This is challenging in multiple aspects for health care professionals and leads to increased usage of health care resources [2][3]. This leads to more complicated and/or multiple concurrent treatments, which usually lead to long-term use of multiple drugs in combination [4][5], called polypharmacy. Countermeasures to issues arising from polypharmacy are urgently required, as polypharmacy is common in elderly patients [6] and home hospitalization requires systems to monitor possible complications to ensure patient safety and provide a better medication prescription. Therefore in this work, we focused on potential drug-drug interactions (Potentially Inappropriate Prescribing, PIPs), and developed a rule-based system to detect possible adverse events.

Polypharmacy is defined as the concurrent use of multiple (usually more than four) medications or, sometimes, as the unnecessary use of multiple and/or redundant medications [7]. As mentioned before, this is common in adults older than 65 years, which shows that generally more than half of all patients older than 65 years take more than 5 prescription drugs [6]. The situation is complicated further by over-the-counter medications. Studies regarding such medications show that, especially in certain communities, 90% of the patients take more than 1 and almost 50% take 2 to 4 of these freely available medications [6][8]. Additionally, because of incomplete case histories and cases of low patient compliance, the medical professionals treating the patient often have incomplete knowledge on which substances the patient is actually using. Patient safety is a problem area and topic of active research in general, as adverse drug events are a serious problem in modern health care. Multiple studies brought this to attention, notably the report "To Err is Human" in the US, however, adverse events are preventable in many cases [9][10][11].

Multiple clinical guidelines and screening tools have been developed to check for PIPs. Mark Beers et al. created a list of medications that can be considered inappropriate for older patients in long-term care in 1991 [12]. Beers' criteria were updated regularly and are the basis for other criteria sets, most notably "Screening Tool of Older Persons potentially inappropriate Prescriptions" (STOPP) and "Screening Tool to Alert doctors to the Right Treatment" (START). Both are evidence-based lists of criteria, first published in 2008 and developed in Ireland by a round of experts using the Delphi consensus method [13][14]. Version 2 of these criteria was published in 2014 [15]. STOPP/START resulted in much research interest, many countries and institutions support the tools and consider them appropriate for evaluating prescriptions [16]. Here is an example of the STOPP criteria: The following prescriptions are potentially inappropriate to use in patients aged 65 years and older for cardiovascular system:

1) Digoxin for heart failure with normal systolic ventricular function (no clear evidence of benefit).
2) Verapamil or diltiazem with NYHA Class III or IV heart failure (may worsen heart failure).
3) Beta-blocker in combination with verapamil or diltiazem (risk of heart block).

And here is an example of the START criteria for the respiratory system:

1) Regular inhaled β 2 agonist or antimuscarinic bronchodilator (e.g. ipratropium, tiotropium) for mild to moderate asthma or COPD.
2) Regular inhaled corticosteroid for moderate-severe asthma or COPD, where FEV1 <50
3) Home continuous oxygen with documented chronic hypoxaemia (i.e. pO2 <8.0 kPa or 60 mmHg or SaO2 <89

However, none of these guidelines were available in a machine readable format, they are indented to be used manually by medical professionals, which can create a considerable workload. The usage of the guidelines out of paper documents is likely unrealistic due to time restrictions of the medical staff. There is an urgent need to translate such rules into machine readable form and integrate them into decision support system as part of the medication prescription process making them in almost real time available for medical doctors. So in the frame of this work a set of machine readable rules was created using the Drools rule engine [17]. The target of this work was to translate the STOPP/START guidelines into machine readable rules using the Drools format, wrapped them in a prototype service application. The developed prototype is being integrated with the home monitoring system allowing its extension beyond existing systems' capabilities and approaches especially when combined with real time sensors' data measuring the body vital signs and utilizing machine learning algorithms. production rule systems have been used together with other technologies in ”business rule management systems” since the early 1990s, especially in industries with a lot of rules in everyday operation, such as insurances [18]. The RETE algorithm that optimizes the process of matching conditions to rules by ”compiling” a network of conditions and their relation has been designed in the 1980s for such scenarios [19]. Derivatives and improvements of this algorithm are still used in current rule engines, such as Drools [20]. The usage of a decision support system to reduce medication errors shows good results, especially when used at the ordering stage of a medication [21]. A Business Rule Management System (BRMS) is software that creates, supports, and executes decision logic and business rules. Drools is one of the most used BRMSs being utilized by thousands of organizations currently. The object-oriented system is an augmented implementation of the known Rete algorithm tailored for the Java language. It includes both forward as well as backward chaining interference based rules engine and it provides a framework to allow business logic externalization in a common place. Efforts have been made to compute sets of medical guidelines for using them in applications. One example is STRIPA [22], a rule-based decision support system for medication reviews. It was developed with the Systematic Tool to Reduce Inappropriate Prescribing (STRIP) in mind, a drug optimization process, and aims at making the pharmacotherapeutic analysis step easier and less time-consuming by automation [22]. This system was not targeted to integration in home monitoring systems and was developed as a stand alone system.

In section 2, an elaboration on design principles, system design and involved frameworks is given. In section 3, the design implications, data model and the approach of implementation are described. In section 4, the evaluation methodology and the results are depicted. In section 5, conclusions and future work are discussed.

II. Design

This prototype was designed with usability focus, to have a structured and easily manageable representation of the rules, without losing too much precision in detecting rule violations or losing too much flexibility in the addition of rule conditions and the manipulation of rules. With this in mind, we looked at state-of-the-art rule engines with a wide implementation in the industry, such as Drools. Additionally, with the use of the Drools Rule Language (DRL), rules can also be generated from schematic representations, so called decision tables. This confirmed our choice of Drools as the core of the system, as it has also proven itself in similar applications [22].

A. System Design

As shown in Figure 1, the system was designed as a self-contained service, with two possibilities for interoperability with the rest of the home monitoring system in mind. One is the usage of the same database and framework as other related projects, Apache Cassandra and Apache Spark, so that the new service can be introduced into a new environment without changing much of the code and/or configuration. The second possibility is a REST API (via KIEServer) that can be used for sending data to the service for evaluation against the rule base and is designed to be easily extended. During development, the focus was set on the first possibility: the system was built and tested with the same tools and technologies that allow easy integration into home monitoring systems.

It is important to note that Drools is used in a stateless fashion. Stateless Drools sessions can be called like a function, a batch of data is passed to the session and the results of the rule execution are sent back. The production rule system does not keep track of (generated) knowledge and the result of one rule execution will never trigger or influence the execution of other rules. This was the desired operation mode in the use case at hand.

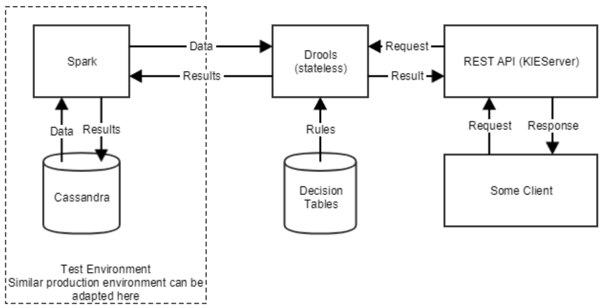


Fig. 1: Overview system design

B. Analysis of the Guideline Structure

In the early phase of the development, the STOPP and START sets of criteria were analyzed for their structure. As an intermediate result, the following parts could be identified in most STOPP rules:

- The rule subject, being a drug or drug family
- A part specifying co-medication that might interact with the subject
- Therapeutic information / information on pharmacotherapy, filtering for special (mis-)use cases of the subject
- A diagnosis / treatment condition, in some cases the subject is only harmful / not harmful if some condition, symptom or treatment is present
- Some ”dependent clinical characteristic”, an additional condition that can be a diagnosis, symptom or lab value for example that narrows down the execution of the rule (often exceptions to the rest of the rule)
- The outcome: in the case of STOPP rules that is a warning to consider a medication change for the patient

START rules have similar parts that can be identified, but they do not contain a subject in the sense above, meaning a drug or drug group that has to be matched with a drug from the patient’s records for the rule to be executed. Instead, they contain a drug that they advice in case the rule is executed successfully, as START aims to recommend initial and/or additional medication that is proven to be beneficial in the case described in the rule.

The various connective words and phrases used in the plain English statements of the original set of criteria could be reduced to ”and”, ”or” and ”not”. Statements containing temporal modifiers like ”X with concurrent Y” were also reduced to the logical operators above. However, at present this first prototype does not have the required structured data from electronic health records to execute that specific type of rules.

The identified rule parts were also further sub-divided into ”types”. Conditions on drug dosage, time of prescription and co-therapy could be reduced to conditions on intolerance, efficacy, duration, dosage and a check for contraindication. Further time/co-therapy conditions were: ”The subject is the first treatment for something”, ”used as a long-term treatment”, ”used as secondary prevention”, ”used when an alternative is available”, ”used as mono-therapy”, ”used instead of some other drug” and ”used with some other drug of the same class”. Similarly, detailed types of symptom / diagnosis conditions were: ”Usage of the subject drug with a history of some diagnosis X”, ”used as a treatment for X”, ”used for therapy in X” and ”used unless concurrent X”. Clinical characteristics could be reduced to: Health conditions, physical examination result, interventions, disease history, laboratory results.

III. Design Implications and Implementation

In principle, it would have been possible to create conditions in the machine readable version of the criteria sets for all of these condition types. However, as such a level of detail was not available in the data and is difficult to achieve and use correctly, many types were left out or combined into very basic conditions. Some information is contained in the codes of the classification systems used and a good selection of codes for condition checks allows representation of some of these detailed condition types in their more general parent condition. With more detailed data, additional conditions can be implemented, but some only appear in very few and quite specific rules, it might be better to accept the possibility of false alerts and let the medical professional decide if acting is necessary, instead of trying to make a rule more precise with complicated conditions on unreliable data. This trade-off had to be evaluated throughout the design and development process. It will also be important for future improvements and extensions. For the realization of the concept, proven technologies have been chosen. The requirements fit the use case of a production rule system. Additionally, a general purpose programming language, such as Java was chosen as a consequence of the choice of the rule engine.

A. Data Model

As Drools is data-driven, the data model is very important. It consists of a typical object-oriented programming class hierarchy: all used objects are plain Java objects. Each type that was used in one of the decision tables is represented by a Java class, which includes objects representing patients, drugs and diagnosis. As shown in Figure 2, the system is currently only using coding systems for both diagnosis and drugs, the presented objects basically only act as a container for these codes with some additional functionality.

The defined objects contain methods for matching codes and code prefixes by simple string matching. This way, the rules can easily take advantage of the structure of the mentioned coding system. For example, if a guideline from the STOPP set states that all opioids should never be given together with some other drug, we can take the common prefix for opioids in the Anatomical Therapeutic Chemical (ATC) system and use it for pattern matching in the rule, by just passing it system for evaluation. This is easier than compiling lists of drugs manually and less error prone, but not as simple as it may sound, in fact, it is quite difficult in many cases to find a

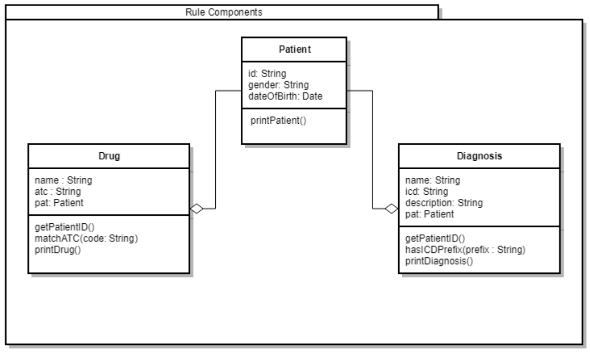


Fig. 2: Overview system design

coded representation of what is stated in the original criteria that means exactly the same semantically. Still, we will see how rules make use of the coding systems in detail in the next sections.

B. Rule Structure and Decision Tables

Each rule can be fitted to a schema consisting of conditions. Basically, as mentioned above, the following parts could be identified as a general structure in all STOPP/START criteria: a rule subject, being a drug or drug set, another drug or drug set, representing drugs that might interact with the subject (co-medication group), a set of diagnosis and a patient, to which all other objects have to relate.

The co-medication group filters for drug-drug interaction using ATC codes, many STOPP rules state something like: ”If the patient is taking drug A and he is also taking drug B at the same time, revoke the prescription for drug B”. The assumption was made that drugs supplied to the system are always prescribed at the same time, there is currently no check for concurrency. This is due to the fact that the data that was available at the time of writing did not include such details. We have examples of how the rule base could be made more precise if the available data is of better quality, something that will become apparent multiple times and has already been mentioned before.

The diagnosis group filters for diagnoses, symptoms, or other information that can be represented by the International Classification of Diseases (ICD) system. This further narrows down the execution of the rule immensely by applying an additional condition that many of the rules have in common. To summarize, the most important criteria and parts of the resulting decision tables are the rule subject, drug interactions, and coded diagnoses, they were the only ones that were selected for implementation.

The observation that most of the rules in both STOPP and START follow a certain scheme lead to the belief that they can be reduced to a fixed structure, basically a prototype rule with parametric conditions. That is why so called decision tables were chosen as the source of rules for the system. Drools supports Microsoft Excel spreadsheets with a certain structure as an input and will generate rules from them. This approach has certain advantages over representing the rules in files using the decision support language described above in certain use cases and the STOPP/START criteria were quite compatible with the approach.

Figure 3 represents a first iteration of a STOPP table. Each rule follows the same structure, as it is given by the decision table. First, the knowledge base is checked for the drug that is the subject of the corresponding STOPP guideline. If the subject drug is found, the following columns contain all other tests, but not always all tests for every rule, some are left out, if the guideline does not contain such a requirement. This can be done by leaving the respective cell empty. The conditions check for interacting drugs, for diagnosis and that the patient ID of both of these objects match the one of the subject drug. We do not verify the patient’s age, although the STOPP/START criteria are designed for people above the age of 65, because the rule base will be used in an environment, where only data of such patients will be processed. An additional column contains the negation of the check for interaction drugs. This was necessary for a few rules, which apply only if a certain drug or drug class is not in the medication plan of a patient. This, of course, leads to the logical conjunction turning into a logical disjunction: the condition requires the medication list of a patient to be free of all the mentioned codes or code prefixes. Note that this generally is a first naive table layout and does not take performance optimization into account. The order of the conditions can be optimized for quicker execution, as we will see, but this always affects all rules, one drawback of using decision tables. If the rule is evaluated to be positive, the marked action of the available ones is taken. In Figure 3, this is one of four very basic output variants. During use of the system later on, this can be any kind if post-processing or event handling.

Decision tables make creating, testing and updating of a larger rule base of similar rules easier. Once the structure of the decision table and the template code is done, only parameters have to be entered into the table. In the case of the rule base presented here, this brings other challenges, mainly in choosing the correct codes for representing the symptoms and illnesses mentioned in the STOPP/START guidelines, but the actual implementation work is reduced. In the case of the ”subject drug” test template code, the rule engine will loop over all parameters found in the rules cell for this condition and that the results of the evaluation of each single one should be connected with the ”or” operator. The parameter is inserted at the placeholder ”$”, in this case we pass it to a function. The template code section allows us to call functions of the object specified in the row for types above. In this case, the function ”matchATC” is called with the parameter inserted in the rule’s cell. We can also see that we can add arrays of parameters to rules by just adding them into the cell, separated with commas, in this case strings representing the ATC code or ATC code prefix we want to match. The method we call is described in the data model section, which we use it to match codes

<table>
<tr><td colspan="7">RuleTable STOPP</td></tr>
<tr><td>NAME</td><td>CONDITION</td><td>CONDITION</td><td>CONDITI</td><td>CONDITION</td><td>CONDITION</td><td>ACTION</td></tr>
<tr><td></td><td>$s : Drug</td><td>$drug : Drug</td><td>$drug : Dı</td><td>$p : Patient</td><td>$d : Diagnosis</td><td></td></tr>
<tr><td></td><td>forall(||){matchATC($)}</td><td>getPatientID() == $s.getPatientID(),forall(||){matchATC($)}</td><td>forall(&&){matchATC($)}</td><td>(id == $s.getPatientID())</td><td>getPatientID() == $p.id, forall(||){$}</td><td>System.out.println("------ STOPP WARNING ------");
System.out.println("$param");
$s.printDrug();
$p.printPatient();
$d.printDiagnosis();
System.out.println("------ END STOPP ------");</td></tr>
<tr><td>NAME</td><td>subject drugs</td><td>complication drugs</td><td>neg. complication drugs</td><td>patient</td><td>diagnosis</td><td>OutputNoDrug</td></tr>
<tr><td>STOPP B1</td><td>"C01AA05"</td><td></td><td></td><td>TRUE</td><td>hasICDPrefix("I50")</td><td>STOPP B1: No clear evidence of benefit</td></tr>
<tr><td>STOPP B2</td><td>"C08DA01","C08DB01"</td><td></td><td></td><td>TRUE</td><td>hasICDPrefix("I50.04"),hasICDPrefix("I50.05"),hasICDPrefix("I50.13"),hasICDPrefix("I50.14")</td><td>STOPP B2: May worsen heart failure</td></tr>
</table>

Fig. 3: The template code of the STOPP decision table

and code prefixes. The template code for the "complication drugs" condition is the same, but adds an equality check for the patient ID to make sure that only drugs that are actually taken by the patient are considered. The template code for the diagnosis check, basically works in the same way. But instead of inserting a parameter into a method call, we just specify that all parameters should be connected in a disjunctive fashion. The rules' fields of this condition check contain both method calls and equality checks. The same result is achieved in a different fashion. This basically concludes the description of the pre-optimization STOPP table.

The decision table for the START criteria works in a similar way as the templates, but has different requirements for the number and order of condition checks and for the patient check. This is because we do not have a subject drug in every case that can be used to get all patients taking the drug like in the STOPP decision table. Instead, we now have to look at all patients. In the next step, all drugs with a matching patient ID are checked against the rule's codes and code prefixes, just as in the STOPP table. The diagnosis check is also the same, the number of parameters is just higher in many cases.

C. Integration into the home monitoring system's workflow

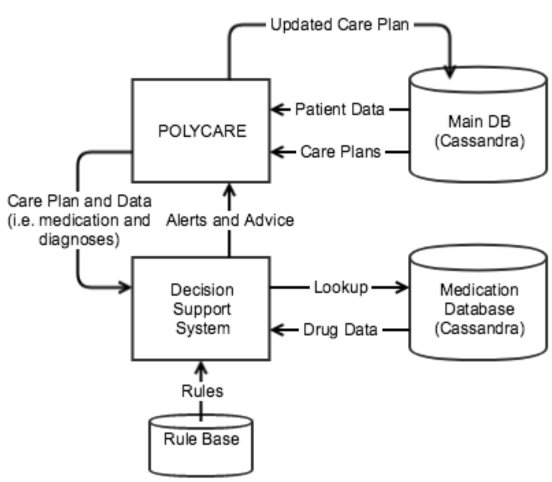


Fig. 4: Overview of integration of the prototype in the home monitoring system Polycare

The system will take in information about a patient and provide feedback about drug interactions and interference while under certain therapies or suffering from certain diseases, as defined in the STOPP/START criteria. Depending on whether a START or STOPP condition is detected, an alert is given or a recommendation for therapy is given. The information is sent back to the subsystem developed for persistence in a database and to be used for care plan creation, as seen in Figure 4. Not shown in the workflow diagram is the process of looking up medications in the Apache Cassandra database. A "Pharmazentralnummer" (PZN, meaning "central pharmaceutical number, a standardized number to identify medications and medical products in Germany) id, from the electronic health record of a patient is used to get detailed information about a drug (more specifically about the active substance) and its classification in other coding systems such as ATC, which is currently used in the system so far.

Finally, information about the medication of the patient and her diagnosis is used. If available, more precise information such as dosage of specific drugs, duration of the treatment and lab values can be used to give better feedback. But in this use case, it is better for the patient if a false alert is raised than if no alert is raised due to incomplete knowledge.

IV. EVALUATION METHODOLOGY AND RESULTS

There are technical measurements and results, such as performance and a possible performance differences after optimizing the initial system and rule base. Additionally, there is a discussion on how the results presented here fit in within the project infrastructure, what drawbacks exist and what may need more work. An issue is the selection of both ATC and especially ICD codes for the rule's conditions. A first selection was used in this work to demonstrate how such a rule base for the STOPP/START criteria and the accompanying system might work, but there was no guarantee for correctness of the rules from a medical standpoint at any time.

Technical evaluation resulted in some key points. Most obviously, as previously mentioned, finding the correct order of condition checks and was part of the task at hand, but turned out to be marginally important for performance of the system. Still, with more conditions and more complex comparisons one should think about the structure of the table again, as it

 ISBN: 978-1-61208-608-8

is important to narrow down the set of possible matches for a rule by applying the simplest conditions that exclude the most facts first. In the case of the START table, there was some performance gain by filtering the diagnosis codes first, as most START rules do not have a drug condition and filtering by diagnosis can exclude a case early on. This also applies to choosing the right starting condition for the rule. In the case of the START table, we unfortunately have to look at the patients in all cases to see if there is a beneficial prescription for them. But the STOPP table can match a drug a patient is taking in the first step. If the data set sent to the knowledge base does not contain a matching code, it can be discarded immediately.

It is important to note that the knowledge base had to be created to work in a stateless fashion: the working memory does not keep track of facts other than initial ones created at the time of starting the system. Incoming data is processed, rules are matched and executed and some action is taken. Afterwards, the data is discarded and the knowledge base remains unchanged. This also means that we can only look at one (the system could be configured to cache data and process batches) set of data representing a patient's record at the time. If the knowledge base were to be created to work with stateful execution, optimizing the rule base and the accompanying system would be a completely different task.

The use of stateless sessions also has some implications for the correct usage of the system. All data belonging to one patient should be inserted and executed in the same batch or at least all the patient's data from the same time frame. Otherwise, it would be possible, for example, to miss a drug-drug interaction as the system does not keep track of knowledge inserted or created during runtime. As an example, if the patient is taking a beta-blocker drug, we know this because of some report and an entry for this was created in a database, but the patient is also taking Verapamil and a separate entry was created for this, it might be possible that we miss the interaction between the two drugs, if there is no pre-processing step that makes sure that all entries for one patient and in the same time frame are collected before sending the data to the rule engine. This has to be kept in mind to get good results from the system.

As mentioned briefly beforehand, the system's performance in terms of accuracy, is determined by the choice of codes to represent the conditions of the original guideline statement. This is especially true, as the technical correctness of the rules was verified using unit testing, which covered nearly all of the cases which might trigger a rule evaluation. But the correctness of the result in a medical sense, of course, still depends on the codes used for the condition checks. The diagnosis codes used in this were chosen by looking at the ICD-10-CM index and choosing codes that seemed suitable. Choosing ATC codes was easier, as it is pretty clear in most cases which substances are mentioned in the original guideline statements. But as the ATC structure allows one substance to have multiple codes if it can be used for different treatment goals on different physiological systems, one has to choose the correct code or include all variants. Generally, it is always possible to include more codes, especially in the case of diagnosis codes, to catch more rule violations. But this can lead to more false alarms. Working around the coding systems by, e.g., compiling a list of single codes instead of using the hierarchy of the system, is generally more difficult and inefficient than using well-defined codes or the structure of the coding system, but sometimes there is no other way.

Some rules from the STOPP/START sets were especially difficult to represent by an entry in the respective decision table. Rule STOPP D10 for example, warns that neuroleptics should not be used as hypnotics. As there is no ATC group that contains all neuroleptics, a list of codes had to be created manually. Modafinil, sodium oxybate and methylphenidate were used as the codes for the subject drug check. But this must be confirmed and corrected by medical professionals yet. More difficult and quite representative of the issues in representing the criteria as formal rules is the condition that these neuroleptic medications must be stopped if they are used as hypnotics. There is no information about the treatment goals or the intent of the medication available in the test data sets, and this information will also not be available for the prototype development. To catch all cases that might put the patient in danger, a compromise had to be made: within our first implementation, all neuroleptics will trigger rule STOPP D10, even if their usage is correct and not dangerous to the patient. The physician that receives warnings from the system will have to decide, whether the warning is correct or just a false alarm. Similar decisions had to be made for most rules that had some kind of exception or modifier in their original statement.

Most recommendation systems and similar applications have to find a balance regarding this issue. In our case, if the number of false alarms is too high, the actual value of having an automatic system for evaluation of the STOPP/START guidelines is reduced, as the amount of work compared to manually checking the criteria is not small enough to justify using the system. Similarly, if the number of cases missed by the system that actually put the patient in danger is too large, there is also no benefit in using it.

The rule base and system presented here will be evaluated and refined with these criteria in mind in the future, after integration together with the rest of the home monitoring infrastructure. The used codes will need to be checked again. There already have been panels [23] that worked on finding codes for STOPP/START and they can be used together with new consensus finding methods and a group of medical experts to agree on a set of codes for the rules. A study on the acceptance of the whole system by medical professionals and patients can further show if the selection of codes and the knowledge base in general is too strict or too inaccurate. Feedback and experience with the system in a productive environment might even be more efficient than the theoretical selection of codes and conditions.

V. Conclusion and Future Work

There are multiple things to take away here and that left an impression on us during the development of the rule base. First, there is a big selection of technologies and software freely available for such a task, with active development and very prominent credentials. Their features were not really used to the fullest extend, but as the choice matches what is used in other parts of home monitoring systems, one can say that scaling the project will not be a problem. Drools also seems to be a good choice, it can create or read rules from multiple sources with flexible and powerful template coding and accompanying tools that can be used to modify rules even by people who do not know a lot about programming. It allows the use of the knowledge base as a server, but can also be integrated into any other form of Java application.

Using decision tables to represent the STOPP/START criteria was the correct choice. The criteria turned out to be surprisingly similar in their structure, allowing the use of template code to generate rules. Drools is also quite efficient in matching and executing rules. Running both the 77 rule STOPP and 34 rule START set against 10.000 test records took less than 1 second per case on a consumer grade laptop. However, in the case at hand, changing the structure of a table, for example the order of the rule's condition, did not improve performance significantly. In production use, it would be advisable to focus on the efficient usage of Apache Spark and Apache Cassandra, as the code for database access, data manipulation and analysis will impact performance the most.

While the decision tables are not too complex, the complexity of the problem lies in choosing the correct codes for both drugs and diagnosis. This is not a trivial problem and it impacts the whole domain. As many other authors have stated in their papers on similar projects and on electronic health records, standards for medical data, improved standards for coding and agreements on translating between them and from natural language would help the whole domain immensely. The rules that were created during the development of this work were not evaluated for their "medical correctness". We have seen that choosing codes that precisely represent what was stated in the STOPP/START criteria is difficult. The original statements are sometimes subject to interpretation itself. For any future development, it would be advisable to have a consensus process with medical experts to agree on coding, as many other projects did. Until there is an "official" coded version of the original criteria, there is no other possibility if coded data is to be used. One of the main conclusions drawn from studies [24] is that the quality of the available data is one of the biggest factors in the success of using such guidelines with a decision support system.

With these limitations and problems in mind, the presented rule base can still be a starting point for the decision support system. It will certainly be improved and evaluated further in the next development rounds. Many extensions are conceivable, for example the use of the Resource Description Framework (RDF) for representing rules built from conditions was one idea that came up during development. Giving feedback to the used standards and contributing to them is also important, in this case a Fast Healthcare Interoperable Resources (FHIR) resource could be devised for medical alerts. Another extension could consist of additional conditions to make rules more precise. Lab values, for example, are mentioned in certain criteria of the STOPP/START criteria and could be compared to lab values from a data source. But again, semantic equality has to be ensured, a variety of abbreviations and codes for lab values have to be translated to match the data source. Instead of relying on coded data, one could also make use of a natural language processing system that matches certain terms from written reports of the patient and can also infer the context in which some information is stated. Natural language processing systems are not new to the domain and have shown promising results, surpassing the use of coding in sensitivity [25]. Such systems can be a viable alternative to the use of administrative data and codes.

Finally, it is evident that the quality of underlying data and the systems used to structure it determine what can be done on top of them. The best decision support system in the world can only be as good as the input data it works on, even if methods to further infer knowledge are used. In general, as the quality of health care data increases and as more processes, data formats, workflows and other technologies are standardized, it will become easier to build systems giving warning and advice regarding medication that work with data from electronic health records.

Acknowledgment

This work was partially funded by the European Commission H2020 Programme (Grant Agreement 690367, POLYCARE: Poly-stakeholders integrated care for chronic patients in acute phases). The authors would like to acknowledge the support of the POLYCARE consortium for this work.

References

[1] World Health Organization, "Global health and aging," Geneva: World Health Organization, 2011.

[2] B. Starfield, K. W. Lemke, R. Herbert, W. D. Pavlovich, and G. Anderson, "Comorbidity and the use of primary care and specialist care in the elderly," The Annals of Family Medicine, vol. 3, no. 3, 2005, pp. 215–222.

[3] T. Lehnert et al., "Health care utilization and costs of elderly persons with multiple chronic conditions," Medical Care Research and Review, vol. 68, no. 4, 2011, pp. 387–420.

[4] K. Moore, W. Boscardin, M. Steinman, and J. B. Schwartz, "Patterns of chronic co-morbid medical conditions in older residents of us nursing homes: differences between the sexes and across the agespan," The journal of nutrition, health & aging, vol. 18, no. 4, 2014, pp. 429–436.

[5] G. Kojima, "Prevalence of frailty in nursing homes: A systematic review and meta-analysis," Journal of the American Medical Directors Association, vol. 16, no. 11, 2015, pp. 940–945.

[6] E. R. Hajjar, A. C. Cafiero, and J. T. Hanlon, "Polypharmacy in elderly patients," The American journal of geriatric pharmacotherapy, vol. 5, no. 4, 2007, pp. 345–351.

[7] S. P. Stawicki and A. Gerlach, "Polypharmacy and medication errors: Stop, listen, look, and analyze," Opus, vol. 12, 2009, pp. 6–10.

 ISBN: 978-1-61208-608-8

[8] S. I. Haider, K. Johnell, G. R. Weitoft, M. Thorslund, and J. Fastbom, "The influence of educational level on polypharmacy and inappropriate drug use: A register-based study of more than 600,000 older people," Journal of the American Geriatrics Society, vol. 57, no. 1, 2009, pp. 62–69.
[9] E. C. Davies et al., "Adverse drug reactions in hospital in-patients: a prospective analysis of 3695 patient-episodes," PLoS one, vol. 4, no. 2, 2009, p. e4439.
[10] C. Kongkaew, P. R. Noyce, and D. M. Ashcroft, "Hospital admissions associated with adverse drug reactions: a systematic review of prospective observational studies," Annals of Pharmacotherapy, vol. 42, no. 7-8, 2008, pp. 1017–1025.
[11] J. Lazarou, B. H. Pomeranz, and P. N. Corey, "Incidence of adverse drug reactions in hospitalized patients: a meta-analysis of prospective studies," Jama, vol. 279, no. 15, 1998, pp. 1200–1205.
[12] M. H. Beers et al., "Explicit criteria for determining inappropriate medication use in nursing home residents," Archives of internal medicine, vol. 151, no. 9, 1991, pp. 1825–1832.
[13] P. Barry, P. Gallagher, C. Ryan, and D. O'Mahony, "START (screening tool to alert doctors to the right treatment)—an evidence-based screening tool to detect prescribing omissions in elderly patients," Age and ageing, vol. 36, no. 6, 2007, pp. 632–638.
[14] P. Gallagher, C. Ryan, S. Byrne, J. Kennedy, and D. O'Mahony, "STOPP (screening tool of older person's prescriptions) and START (screening tool to alert doctors to right treatment). consensus validation." Int J Clin Pharmacol Ther, vol. 46, no. 2, Feb 2008, pp. 72–83.
[15] D. O'Mahony et al., "STOPP/START criteria for potentially inappropriate prescribing in older people: version 2," Age and Ageing, vol. 44, no. 2, 2015, p. 213. [Online]. Available: http://dx.doi.org/10.1093/ageing/afu145
[16] B. Hill-Taylor et al., "Application of the stopp/start criteria: a systematic review of the prevalence of potentially inappropriate prescribing in older adults, and evidence of clinical, humanistic and economic impact," Journal of Clinical Pharmacy and Therapeutics, vol. 38, no. 5, 2013, pp. 360–372. [Online]. Available: http://dx.doi.org/10.1111/jcpt.12059
[17] D. Project, "Drools documentation," 2017, [retrieved: January, 2018]. [Online]. Available: http://docs.jboss.org/drools/release/latest/drools-docs/html_single/#d0e32
[18] S. Hildreth, "Rounding up business rules," Computerworld, vol. 39, no. 21, 2005, p. 25.
[19] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," Artificial Intelligence, vol. 19, no. 1, 1982, pp. 17 – 37. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370282900200
[20] M. Proctor, "R.I.P. RETE time to get PHREAKY," 2017, [retrieved: January, 2018]. [Online]. Available: http://blog.athico.com/2013/11/rip-rete-time-to-get-phreaky.html
[21] R. Kaushal, K. G. Shojania, and D. W. Bates, "Effects of computerized physician order entry and clinical decision support systems on medication safety: a systematic review," Archives of internal medicine, vol. 163, no. 12, 2003, pp. 1409–1416.
[22] M. C. Meulendijk et al., "Computerized decision support improves medication review effectiveness: An experiment evaluating the STRIP assistant's usability," Drugs & Aging, vol. 32, no. 6, 2015, pp. 495–503. [Online]. Available: http://dx.doi.org/10.1007/s40266-015-0270-0
[23] D. A. de Groot et al., "Specifying icd9, icpc and atc codes for the STOPP/START criteria: a multidisciplinary consensus panel," Age and Ageing, vol. 43, no. 6, 2014, pp. 773–778. [Online]. Available: http://dx.doi.org/10.1093/ageing/afu075
[24] M. Price, I. Davies, R. Rusk, M. Lesperance, and J. Weber, "Applying STOPP Guidelines in Primary Care Through Electronic Medical Record Decision Support: Randomized Control Trial Highlighting the Importance of Data Quality," JMIR Med Inform, vol. 5, no. 2, Jun 2017, p. e15.
[25] H. J. Murff et al., "Automated identification of postoperative complications within an electronic medical record using natural language processing," JAMA, vol. 306, no. 8, 2011, pp. 848–855. [Online]. Available: http://dx.doi.org/10.1001/jama.2011.1204

Edge Computing Architectures – A Survey on Convergence of Solutions

Eugen Borcoci, Serban Obreja
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
Emails: eugen.borcoci@elcom.pub.ro, serban@radio.pub.ro

***Abstract* —Edge computing architectures and technologies, recently proposed, are complementary to centralized Cloud Computing, given that edge computing can offer faster response, higher context and location awareness, better mobility features, minimization of the data transfer to the centralized data centers, flexibility and so on, for a large scale of applications, including Internet of Things. Several approaches have been developed in parallel by different entities, like research groups, industry, operators, and standardization organizations. Fog computing, Multi-access (Mobile) Edge Computing, cloudlets, etc., are relevant examples, included in the large class of Edge computing. Their architectures and technologies have many essential common characteristics, but also differences in approach. A natural question is raised - if any significant convergence (which is not yet seen) will emerge in the near future. This paper is not intended to be a complete survey, but it attempts to identify some convergence directions and issues related to the Edge computing technologies.**

Keywords — Edge computing; Mobile (multi-access) Edge Computing; Fog computing; Software Defined Networking; Network Function Virtualization.

I. INTRODUCTION

The significant growth of the Internet based applications and services contributed to a steep rise in data storage and processing requirements. Cloud Computing (CC) is a powerful solution in this context, by integrating the advancements in computing and network technologies. The Cloud Computing paradigm is mainly based on the data centers which are capable of handling storage and processing of large amount of data. The data centers can be interconnected over optical networks to form data center networks (DCNs), seen by the end user as a unique powerful resource.

Internet of Things (IoT), mobile applications and services, smart cities applications, etc., recently emerging, pose novel challenging requirements the CC based solutions. Cloud computing centralization (in terms of processing and storage) in traditional data centers have inherent limitations, being non-appropriate for the above mentioned specific classes of applications. The IoT, mobile applications, vehicular networks et al., require real or near real-time response/latency, high bandwidth, location and context awareness, reduction in amount of data transferred to CC and back, more flexibility in functional distribution, resource usage optimization and others.

One solution for the above problems is Edge-oriented computing (EC) [1][2], where the main idea is to create additional CC capabilities at the network edge, close to, or even installed in the “terminal” data sources (e.g., vehicles). EC is not seen as a replacement of the CC but it is complementary to it. The CC and EC can be used independently or in cooperation. Several proposals currently exist for EC, like: *Fog Computing* (FC), *Multi-access (Mobile) Edge Computing*, *Cloudlets*, etc.

A current open research issue is to investigate if any convergence could be established in terms of concepts, architecture and implementation of such solutions.

Edge computing is of high interest not only in research communities; there are several entities involved and active in EC, in industry, research and standardization organizations. Different sets of specifications have been elaborated by independent entities.

Fog Computing (FC) is a term coined by CISCO (2012) [3][4] primarily to serve IoT needs.

Later, the *OpenFog Consortium* organization (November 2015) [5][6] has been created, having as founders: Cisco, ARM, Dell, Intel, Microsoft, Princeton University Edge Laboratory and comprises more than 60 members today. They defined an FC and *Open Reference Architecture* [5] and are creating standards to enable interoperability in IoT, 5G, artificial intelligence, tactile Internet, Virtual Reality (VR) and other complex data and network intensive applications.

The European Telecommunications Standardization Institute (ETSI) established in 2014 the *Mobile Edge Computing Industry Specification Group* [7][8]; in March 2017, the name has been re-defined as: *Multi-access Edge Computing ISG* [9] to include also non-cellular operators requirements. Cooperation started recently between Open Fog Consortium and ETSI MEC ISG to give the industry a cohesive set of standards around Fog Computing in mobile environments, while eliminating redundancy.

Another entity is *Edge Computing Consortium (*Dec. 2016) was formed, having as initial founders Huawei, Intel, ARM, and now comprising several members. The *US National Institute for Science and Technology (NIST) elaborated a* definition of FC (draft closed in September 2017) [10]. Also IETF recently started to contribute to the development of these technologies.

At Carnegie Mellon University, they developed the *Cloudlet* [11][12]; a cloudlet is middle tier of a 3-tier hierarchy: “mobile device – cloudlet – cloud”. It can be seen as a "data center in a box" whose goal is to "bring the cloud closer" to the data sources. Cloudlets are mobility-enhanced *micro data centers* located at the edge of a network and serve the mobile or smart device portion of the network. They are designed to handle resource-intensive mobile applications and take the load off both the network and the centralized data center and keep computing close to the point of origin of information.

Microsoft Research proposed in 2015 [13] the *Micro data center*, as an extension of today's hyperscale cloud data centers (e.g., Microsoft's Azure). The goal is to meet new requirements, e.g., lower latency and new demands related to devices (e.g., lower battery consumption).

The EC can generally represent any set of computing and network resources along the path between data sources and cloud data centers. However, *there is not yet a unique vision on "edge" semantics*, except the common attribute of proximity of the EC capabilities to the data sources. An important overlap exists between particular EC architectures; so, convergence is predicted in the near future.

The EC deployment can be strongly helped by novel software technologies. In the architectural management and control planes, *Software-defined networking* (SDN) [14] and *Network Function Virtualization* (NFV) [15][16] are seen as a strong support, given their features like flexibility, programmability, abstraction via virtualization, dynamicity, etc.

This paper is organized as follows. Section II summarizes the use cases and applications for which Edge computing is an attractive technology. Section III is a short overview of Edge computing architectures. Section IV identifies some common characteristics in different EC approaches, while emphasizing some points of convergence. Section VI contains conclusions and possible future work.

II. EDGE COMPUTING BASED APPLICATIONS AND USE CASES

This section gives examples of domains where EC (Fog, MEC, Cloudlets, etc.) could be effective to support the applications [1-4][7][9][12][17-19]. There is a need of open architectures based on EC, to enable interoperability in various domain of applications like in IoT, Artificial Intelligence, novel generation of networking 5G, tactile Internet, vehicular networks and Internet of Vehicles (IoV), smart cities applications, services virtual reality, and other complex data and network intensive applications. In particular, IoT applications generate high amounts of data that can be useful in many ways. Therefore, EC nodes can be used to carry out data mining and data analysis on a large volume of multi-modal and heterogeneous data from various sensor devices and other IoT devices to achieve real time and fast processing for decision making.

The Fog architecture can be hierarchically organized. This can be useful big data analysis in smart cities. Experimental results demonstrated the feasibility of the system's city-wide implementation in future smart cities scenario. Fog computing can support new services for mobile networks requiring high data rates and low latency (e.g., virtual reality).

Internet of vehicles [18] may benefit from EC (e.g., Fog, MEC) capabilities to develop a large range of applications like: safety and management-oriented (traffic safety, traffic and navigation management, remote telematics); business-oriented (infotainment, insurance car sharing, etc.). The vehicle itself can be equipped as to become a fog node, to attain optimum utilization and benefit from vehicular communications and computational resources. The mobile fog nodes can inter-communicate and provide services including infotainment, advanced driver assistance systems, autonomous driving, collision avoidance, and navigation.

Other area of applications is oriented to emergency, health care services. The latency-sensitive and security-privacy-sensitive services also can benefit from fog/edge nodes capabilities. Experimental results validated that EC supporting cyber-physical systems can improve the cost efficiency significantly based on by jointly considering base station association, task distribution.

MEC technology is a particular EC case where the MEC resources (i.e., MEC servers) are placed at the network edge (e.g., in *Radio Access Network* (RAN), i.e., Base Stations, or in aggregation points, etc.). It offers low latency, proximity, high bandwidth, and real-time insight into radio network information and location awareness. Therefore MEC can support many applications and services for multiple sectors, (enterprise, consumer, health, vehicular, etc.) [7].

In *RAN-aware Content Optimization*, an application running in the MEC server, can expose accurate cell and subscriber radio interface information (cell load, link quality) to a content optimizer, enabling dynamic content optimization, improving Quality of Experience (QoE) perceived by the users and improve network efficiency. Dynamic content optimization enhances video delivery through reduced stalling, reduced time-to-start and 'best' video quality. Among smart cities applications, a video streaming service can benefit from MEC approach. Video streams from monitoring devices can be locally processed and analyzed at the MEC server to extract meaningful data from video streams. The valuable data can be transmitted to the application server to reduce core network traffic. Other mobile application where MEC could be a subsystem is *Augmented Reality* (AR); this requires low latency and a high rate of data processing in order to provide the correct information depending on the location of the user. Collaboration is required for data collection in the uplink, computing at the edge, and data delivery in the downlink. The data are actually processed in a local MEC server rather to improve the user experience.

MEC is also useful in IoT, given that IoT devices often have low capabilities (processing, storage capacity). There is a need to aggregate various IoT messages connected through the mobile network close to the devices. Gateways (collocated with MEC servers) aggregate the messages and ensure security and low latency. To achieve an efficient service, grouping of sensors and devices is accomplished.

This approach also provides an analytics processing capability and a low latency response time.

III. EDGE COMPUTING ARCHITECTURES

A summary of some EC relevant architectures is presented in this section. Note that this paper investigates a possible convergence between EC architectures and technologies. In this respect, is to be noted that, currently, *there is no unique vision neither on the terminology, nor on architectural definition for EC.*

We selected and give below a summary of the *Fog computing* and *Mobile Edge Computing* and architectures, which seem to be of strong interest for industry, operators, and standardization organizations.

Fog computing (FC) is an important EC technology complementary to CC (i.e., cooperation CC/FC is envisaged). The FC distributed platform brings computation close to its data sources, to reduce the latency and cost of delivering data to a remote cloud. FC has been proposed originally to support the IoT, introduced by Cisco (Bonomi [3]]).

The *OpenFog Consortium* (2015) [5] defines FC as a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking *anywhere along the continuum* from a cloud data center down to things. Therefore, FC extends the traditional CC model; implementations of the architecture can reside in multiple layers of a network's topology. The CC benefits are extended to FC (containerization, virtualization, orchestration, manageability, and efficiency) and FC can cooperate with CC. OpenFog reference architecture includes security, scalability, openness, autonomy, RAS (reliability, availability and serviceability), agility, hierarchy, and programmability. The FC focuses the processing efforts outside the cloud data center, i.e., in the fog area. Data are gathered, processed, and stored within the network, by way of an IoT gateway (GW) or a FC node (FN). Information is transmitted to this GW from various sources and it is processed in FN; then, relevant data (plus additional command - if necessary), are transmitted back, towards the devices. A FN can process data received from multiple end-points and send information exactly where it is needed.

Note that OpenFog Consortium sees the EC differently from FC, in the sense that FC works with the cloud, whereas EC is defined by the exclusion of cloud. FC is hierarchical, where EC tends to be limited to a small number of layers. In addition to computation, FC also addresses networking, storage, control and acceleration.

This vision is not agreed by all documents related to Edge- oriented computing.

On the other part, MEC, originally targets only the very edge part of the network (e.g., RAN). FC can support multiple industry verticals and application domains delivering intelligence and services to users and business. FC capability is spanning across multiple protocol layers and is not dependent on specific access systems.

OpenFog Consortium defined a *flexible deployment* hierarchical model for FC, IoT-oriented [5]. The model is mapped on a layered architecture consisting of the following layers: 1. *Sensors and actuators* (bottom layer); 2. *Monitoring and control*; 3. *Operational support*; 4. *Business support;* 5. *Enterprise systems* (highest layer).

The flexibility of the model consists in the fact that, depending on the nature and requirements of the target application class and, depending on the CC availability, cost, etc., a cooperating (for a combination CC/FC) vertical chain can be defined for layers, 2, 3, 4, e.g.:

L2.FC, L3.FC, L4.FC, or
L2.FC, L3.FC, L4.CC, or
L2.FC, L3.CC, L4.CC, or
L2.CC, L3.CC, L4.CC.

Fog computing targets quite a large range of applications. Therefore, no unique "universal" architecture exists. In terms of Fog architecture, many studies split this problem in two classes (for detailed discussion, see the survey by Mouradian, et. al. [20]):

End-User Application agnostic architectures (comprising End-User Application provisioning, Resource management, Communication functions, Cloud and federation). This class envisages some general aspects, not specific to a given application.

Application specific architectures (Smart living and Smart Cities, Connected vehicles and IoV, Healthcare and other applications).

In [20], also some criteria for evaluation of the architectures are proposed:

Heterogeneity (C1): it should be considered when deciding which application component(s) should be deployed and where;

QoS Management (C2): there are necessary architectural modules for QoS management (e.g., to assure latency) such as migration engine;

Scalability (C3): modules are needed to assure horizontal scalability (e.g., elasticity engine);

Mobility (C4): a mobility engine is necessary to ensure the continuity of a service for the end-user;

Federation (C5): cooperation is needed between different providers in order to ensure the proper coordination of the necessary interactions between application components;

Interoperability (C6): there is a need for appropriate signaling and control interfaces, and appropriate data interfaces to enable interoperability.

In [4], a general hierarchical FC-CC layered architecture is defined (application agnostic). Note that this architecture includes also the CC. Three macro-layers are defined.

Terminal layer is closest to the end user and physical environment. It consists of IoT devices (e.g., sensors, mobile phones, smart vehicles, smart cards, readers, and so on) widely geographically distributed. Note that some devices like mobile phones and smart vehicles having sufficient computing power, can be included in the next Fog layer. They are responsible for sensing the feature data of physical objects or events and transmitting these sensed data to upper layer for processing and storage.

Fog layer is located outside the CC (i.e., in the network) and it is composed of a large number of fog nodes (FN) (routers, gateways, switchers, access points, base stations, specific fog servers, etc.). The FNs can be static, or mobile on a moving carrier and are widely distributed between the end devices and cloud (e.g., cafes, shopping centers, bus terminals, streets, parks, etc.) The end devices can connect with FNs to obtain services. The FNs can compute, transmit and temporarily store the received sensed data. The Fog layer can perform real-time analysis and latency sensitive applications. The FNs are also connected with CC data center by IP core network, and responsible for interaction and cooperation with cloud to obtain more powerful computing and storage capabilities.

Cloud layer includes multiple high performance servers and storage devices, to support for extensive computation analysis and permanently storage of a huge amount of data. It provides various application services, such as smart home, smart transportation, smart factory, etc. However, different from traditional CC architecture, not all computing and storage tasks go through the cloud. According to the demand-load, the cloud core modules are efficiently managed and scheduled by some control strategies to improve utilization of the cloud resources.

In [21], a six-layer FC architecture is presented, comprising the layers described below.

Physical and virtualization layer involves different types of nodes (physical, virtual nodes and virtual sensor networks) distributed geographically. These nodes are managed and maintained according to their types and service demands. The sensors are sensing the surroundings and send the collected data to upper layers via gateways for further processing and filtering.

Monitoring layer supervises the resource utilization, availability of sensors, FNs and network elements. All tasks performed by nodes are monitored in this layer (which node, which task, at what time, what is its output, etc.).

Pre-processing layer performs data management tasks. Collected data are analyzed filtered and trimmed, in order to extract meaningful information. The pre-processed data are then stored temporarily in the temporary storage layer.

Security layer performs the encryption/decryption of data. Additionally, integrity measures may be applied to the data to protect them from tampering.

Transport layer uploads the pre-processed data to the cloud to allow the cloud to extract and create more useful services For efficient power utilization, only a portion of collected data is uploaded to the cloud.

Note that the above architecture does not include the CC itself.

NIST [10] defines the Fog computing as a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud or data centers. FC supports vertically-isolated, latency-sensitive applications by providing ubiquitous, scalable, layered, federated, and distributed computing, storage, and network connectivity. FC has as main characteristics: contextual location awareness, and low latency, geographical distribution with predominance of wireless access, large-scale sensor networks, very large number of nodes, support for mobility, real-time interactions, heterogeneity, interoperability and federation, support for real-time analytics and interplay with the cloud. NIST has defined [10] a three-layer architecture composed of *Smart end-devices* layer, *Fog* layer and *Cloud* layer. The bottom layer of the Fog is named “mist” and comprises an infrastructure close to the end-devices. The so-called “edge” is seen as a part of the Cloud layer.

Despite still different visions on EC semantics, the most agreed vision on EC and FC is that FC is actually a superset of the EC, i.e., FC would include EC.

From the industry world, the Industry 4.0 vision on Fog and IoT emerged [22]. Industrial IoT and Industry 4.0 need for extensive adoption of advanced IT features across multiple Industry verticals. IT and Operational Technology (OT) convergence is aimed. The important step to this aim is the deployment of *Cloud-like resources at the edge and within the Industrial Operational domain.*

FC merges CC features with real-time and safety OT features (efficiency, flexibility and resource management) It applies resource virtualization, real-time and no real-time computing, modern application management, data interoperability middleware, storage, analytics, advanced networking and security. Complementary technologies are *Time-Triggered Technologies* which refers to precise time distribution, time-sensitive networking and computing resource allocation (standardized as *IEEE Time Sensitive Networking TSN);* TSN is a key element of Industry 4.0 and a necessary component of FC in industrial environment. It enables the convergence of Industrial wired protocols towards a unified standard.

Recently, Bacarelli et. al. [23]) extended the FC scope, by defining *Fog of Everything (FoE)* to serve *Internet of Everything (IoE).* The FNs are usually virtualized networked data centers, which run on top of (typically, wireless) *Access Points* (APs), at the edge of the access network, resulting in a *three-tier IoE-Fog-Cloud* hierarchy. In this context, a “thing” (fixed, nomadic or mobile) is a resource-limited user device that needs resource augmentation in order to execute its workload. The work [23] proposes a hierarchical general architecture for a FoE virtualized platform, integrating the building blocks:

- *IoE layer*, where a number of (possibly, heterogeneous) things operate over multiple spatial clusters;
- *Wireless access network (fixed/mobile)*, to supports *Fog-to-Thing* (F2T) and *Thing-to-Fog* (T2F) communication through TCP/IP connections running atop, e.g., *IEEE802.11/15* single-hop links;
- A set of *inter-connected FNs*, that act as virtualized cluster headers;
- *Inter-Fog backbone* (wireline/wireless) providing inter-Fog connectivity and making feasible inter-Fog resource pooling;
- *Virtualization layer*, allowing things to augment their limited resources by exploiting the computing

capability of a corresponding virtual clone. This last runs atop a physical server of the FN that currently serves the cloned thing;

- the resulting *overlay inter-clone virtual network*, that allows P2P inter-clone communication by relying on TCP/IP E2E connections.

The corresponding protocol stack [23] comprises four layers:

IoE layer provides services like: *(a)* T2F access through a reservation-based collision free access protocol for the things served by a same FN; *(b)* F2T broadcast services.

Fog layer performs: *(a)* energy-efficient management of the networking and computing physical resources equipping each FN, and *(b)* energy-efficient management of the inter-Fog traffic conveyed by the wireless backbone.

Overlay layer supports the overlay inter-clone P2P network by: (a) inter-Fog Clone migration; it can be supported by the implementation of the so-called *Follow-Me-Cloud* framework (e.g., Taleb et al., [18]), to solve "live" inter-Fog clone migration, in response to the thing mobility; (b) dynamic management of the required migration bandwidth, to minimize the energy consumed by clone migrations.

Cloud layer orchestrates the overall Cloud-Fog-IoE platform on the basis of the specific features and Quality of Service (QoS) requirements of the running applications. The solutions must be tailored on the expected attributes of the supported applications.

The MEC architecture is another important EC approach. It has been promoted mainly by ETSI [7-9] and offers low latency/response time, high bandwidth, location and context awareness, reduction in amount of data transferred from/to a terminal device to a centralized cloud data center, etc. The ETSI MEC Industry Specification Group (2014) provided first specifications. In 2017, the MEC name (and scope) has been extended to *Multi-access Edge Computing* [9], to include non-cellular and fixed access cases. MEC supports multi-services and multi-tenancy and is usually developed in the operators' networks. However, authorized external third-parties may also make use of the MEC storage and processing capabilities.

The MEC resources are placed at the radio network edge (e.g., in *Radio Access Network* – RAN, i.e., Base Stations, or in aggregation points, etc.). The key element is *MEC application server*, integrated in RAN and providing computing resources, storage capacity, connectivity, and access to user traffic and radio and network information.

The MEC reference architecture is presented in Figure 1 (details, in [8]). The mobile edge host level is the main MEC sub-system, composed of: the *Mobile Edge Host* (MEH) and its *management*. The MEH includes a virtualization infrastructure (based on *Network Function Virtualisation Infrastructure* –NFVI- coming from ETSI NFV framework [15][16]) and the *Mobile Edge Platform* (MEP), supporting the execution of mobile edge applications. The MEC server can be installed in various places at the network edge: at the 4G/LTE macro base station (eNB); at the multi-technology (3G/LTE) cell aggregation site; at the Radio Network Controller (RNC) site, for 3G.

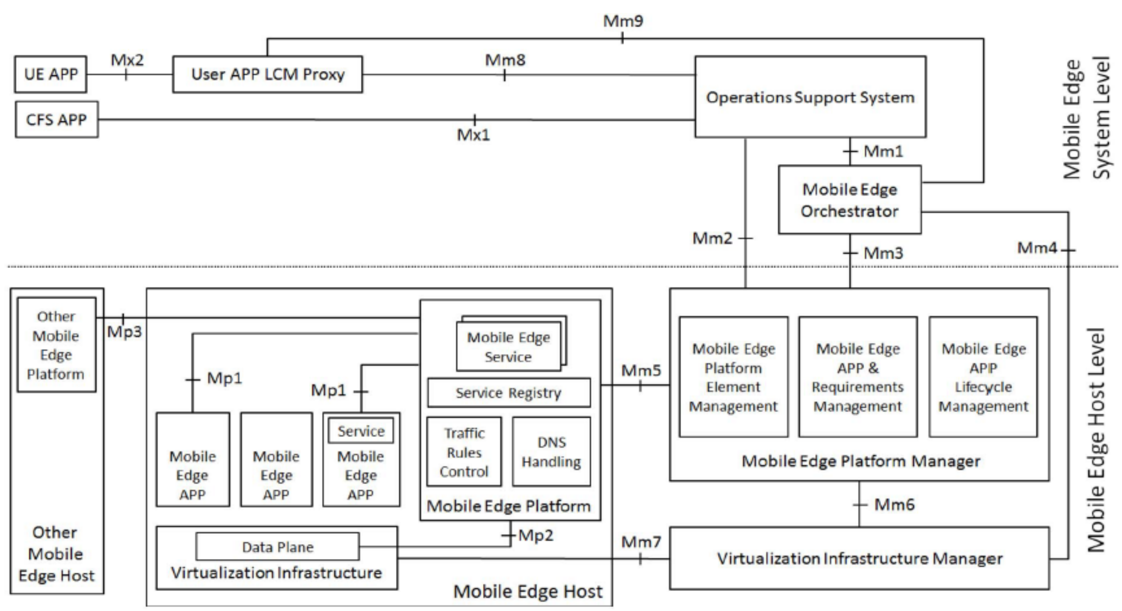


Figure 1. MEC reference architecture (ETSI) [8]

IV. EDGE COMPUTING POSSIBLE CONVERGENCE

The previous section outlined the general characteristics of the EC and summarized two main approaches FC and MEC. This section analyzes some common features and also differences in approaches, to evaluate chances of convergence.

MEC/FC/Cloudlets have quite a lot of common characteristics like: low latency; support for real time interactions, location awareness and mobility and large number of server nodes; geographical distribution proximity to the end devices (single network hop or few hops); service location at the edge of the local network; various working environment outdoor (streets, base stations, etc.) or indoor (houses, cafes, etc.); wireless communication access: WLAN, WiFi, 3G, 4G, ZigBee, etc., or wired communication (part of the IP networks); weak dependence on the quality of core network; low bandwidth costs energy consumption. However, the nodes in FC, MEC, Cloudlets have weak computation and storage capabilities, which raises a need for them to cooperate with CC. All three approaches can benefit from technologies like SDN and NFV in different architectures.

Considering MEC and FC, there are differences between them from several points of view (see [1][6]), as summarised in Table I.

TABLE I. MEC VERSUS FOG DIFFERENCES

Criterion	MEC	Fog computing
Placement of node devices	Servers running in Base stations Network Controller/Macro Base Station	Anywhere -between end devices and cloud data center: Routers, Switches, Access Points, Gateways
Compute Distribution and Load Balancing	Employ a strategy of placing servers, apps or small clouds at the edge	Broader architecture and tools for distributing, orchestrating, managing and securing resources and services across networks.
Software Architecture	Mobile Orchestrator based (strongly specified)	Fog abstraction layer based (only partially specified)
Standardization/ specifications	ETSI/	/OpenFog Consortium
Context awareness	High	Medium
Proximity	One hop	One or multiple hops
Access Mechanisms	Mobile networks: 3G/4G/5G	Wi-Fi, Mobile networks, etc.
Virtualization and management mechanisms	Strongly specified by ETSI (NFV framework)	Larger view of virtualization. In progress at OpenFog Consortium
Hierarchical structure of the overall system	Possible	Yes: multiple levels of cooperating nodes, supporting distributed applications.
Horizontal scalability	Medium	High
Internode Communication	Possible	Supported
Communication with CC	Possible	Required
Modular architecture with multiple access modes	Edge deployments are typically based on gateways with fixed functionality. However they can be made more flexible and dynamic by using NFV	Highly modular HW&SW architecture; every FN is equipped with exactly the resources its applications need; itcan be dynamically configured
Topology of server nodes	Less flexible (limited by RAN spread)	Very flexible

The above table shows several differences between MEC and FC approaches. Note that Table I does not suppose some application specific architectures, but general ones. From this, it is apparent that there is not yet defined a common EC architecture. The MEC/FC/Cloudlets paradigm can offer more or less appropriate support for a large variety applications and use-case scenarios and heterogeneous end devices. On the other side different use cases and applications might have their own set of requirements and trade-offs which can determine which solution is the appropriate choice.

Actually, for a given set of use cases, the selection of an appropriate EC approach is a multi-criteria problem. Among the parameters/criteria for selection those presented in Table I could be considered, if appropriate weights are assigned to them.

However, recently, a strong effort for cooperation started, between different organizations, towards a convergence of vision in the domain of edge computing (including MEC, Fog, Cloudlets, etc.).

Open Edge Computing (OEC) [24][25] is a novel general approach of EC, towards convergence, consisting in small data centers at the network edge, offering computing and storage resources next to the user. Carnegie Mellon University (CMU) performed an early work on Cloudlets at the edge. Given the interests in EC, in 2015 a few parties joined research efforts under the open source banner of Open Edge Computing (OEC). Currently, OEC ecosystem includes CMU, Intel, Huawei, Vodafone, and so on.

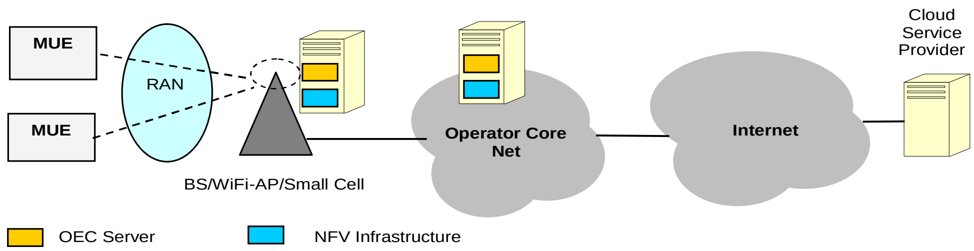


Figure 2. OEC general architecture.
MUE- Mobile User Equipment; BS- Base Station; NFV – Network Function Virtualization

The main OEC goals are: to promote Cloudlets as enabling technology; to drive the necessary technology for various use cases (low latency and computation at the edge) (e.g., extensions to OpenStack, KVM, QEMU); to prototype applications that leverage EC pushing the boundaries and demonstrating benefits; to drive the eco system development for OEC and use current IT solutions.

OEC is engaging with target service industries/sectors through demonstrators and joint projects; with developer communities, seeking feedback and driving EC acceptance. OEC is synchronizing its work with other efforts including ETSI ISG MEC and OPNFV.

The OEC servers can be located close/associated to Base Stations, Access Points, Small Cells, or even in the Operator Core Network (Figure 2). Edge Computing will utilize the Network Function Virtualization (NFV) infrastructure wherever possible. This will reduce significantly the deployment cost of EC.

V. Conclusion and Future Work

This paper presented a preliminary comparative view of some Edge Computing approaches (Fog computing, Mobile Edge Computing, Cloudlets) in order to identify their common and different characteristics and possible chances to have in the near future an EC unified architecture. For the time being, no strong convergence exists, given the large area of use cases and applications targeted to be supported and the pragmatic attempts to tailor the specific architecture to the desired class of applications.

However, recently, a strong effort for cooperation started, between different organizations, towards a convergence of vision in the domain of edge computing (including MEC, Fog, Cloudlets, etc.).

Future work should be done, to investigate more deeply the sets of architectural layers and mechanisms to identify where a common approach can be applied, in order to reduce the development effort and reuse some already developed functional modules.

References

[1] K. Dolui and S.K. Datta, “Comparison of Edge Computing Implementations: Fog Computing, Cloudlet and Mobile EdgeComputing”, Global Internet of Things Summit (GIoTS), 2017, https://ieeexplore.ieee.org/document/8016213/, [Retrieved: May 2018].

[2] Y. Ai, M. Peng and K. Zhang, “Edge computing technologies for Internet of Things: a primer”, Digital Communications and Networks 4 (2018), pp. 77–86, www.keaipublishing.com/en/journals/digital-communications-and-networks/ , [Retrieved: May 2018]

[3] F .Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog Computing and Its Role in the Internet of Things”, August 2012, http://conferences.sigcomm.org/sigcomm/2012/paper/mcc/p13.pdf, [Retrieved: January, 2017].

[4] P. Hu, S. Dhelim, H. Ning and T. Qiu, “Survey on fog computing: architecture, key technologies, applications and open issues”, Journal of Network and Computer Applications 98 (2017) pp. 27–42, www.elsevier.com/locate/jnca, [Retrieved: May 2018].

[5] OpenFog Consortium (2015), OpenFog Reference Architecture for Fog Computing, http://www.openfogconsortium.org/resources/#definition-of-fog-computing, [Retrieved: April 2018].

[6] OpenFog Consortium, “10 ways fog computing extends the edge”, August 21, 2017, https://www.rtinsights.com/10-ways-fog-computing-extends-the-edge/, [Retrieved: May, 2017]

[7] ETSI GS MEC-IEG 004 V1.1.1 (2015-11), “Mobile-Edge Computing (MEC); Service Scenarios”, http://www.etsi.org/deliver/etsi_gs/MEC-IEG/001_099/004/01.01.01_60/gs_MEC-IEG004v010101p.pdf , [Retrieved: January 2018].

[8] ETSI GS MEC 003 V1.1.1 (2016-03), “Mobile edge computing (MEC); Framework and reference architecture,” http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf [Retrieved: January, 2018].

[9] T.Taleb, K.Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration", IEEE Comm. Surveys &Tutorials, vol. 19, no.3, pp. 1657-1681, 2017.

[10] M. Iorga et. al., NIST Special Publication 800-191 (Draft) 1, “The NIST Definition of Fog Computing”, 2017,

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-325.pdf, [Retrieved: January , 2018].

[11] "Cloudlets: all about cloudlet-enabled mobile computing" http://elijah.cs.cmu.edu/, [Retrieved: January , 2018].

[12] Satyanarayan, et. al., "Cloudlets: at the Leading Edge of Mobile-Cloud Convergence", 2014, Int'l Conf.on Mobile Computing, Applications and Services (MobiCASE)http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7026272, [Retrieved: January , 2018].

[13] V. Bahl, " Microsoft, interview about micro datacentres", Sept 2015, http://www.networkworld.com/article/2979570/cloud-computing/microsoft-researcher-why-micro-datacentersreally-matter-to-mobiles-future.html, [Retrieved: February , 2018].

[14] B. N. Astuto, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", Communications Surveys and Tutorials, IEEE Communications Society, (IEEE), 16 (3), pp. 1617 – 1634, 2014.

[15] ETSI GS NFV 002: "Network Functions Virtualisation (NFV); Architectural Framework", http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf, [Retrieved: January , 2018].

[16] ETSI GR MEC 017 V1.1.1 (2018-02), "Mobile Edge Computing (MEC);Deployment of Mobile Edge Computing in an NFV environment", http://www.etsi.org/deliver/etsi_gr/MEC/001_099/017/01.01.01_60/gr_MEC017v010101p.pdf, [Retrieved: January , 2018].

[17] W. Yu, et al., "A Survey on the Edge Computing for the Internet of Things", IEEE Access, 2017, DOI 10.1109/ACCESS.2017.2778504

[18] O. Kaiwartya, A.H. Abdullah, Y. Cao, A. Altameem, and M. Prasad, "Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects" IEEE Access, Special Section on Future Networks, Architectures, Protocols and Applications, vol. 4, pp. 5536-5372, September 2016.

[19] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-Edge Computing for Vehicular Networks- A Promising Network Paradigm with Predictive Off-Loading", IEEE vehicular technology magazine, pp. 36-44, June 2017.

[20] C. Mouradian, et. al., "A Comprehensive Survey on Fog Computing: State-of-the-art and Research Challenges", IEEE Communications Surveys & Tutorials, 2017, https://arxiv.org/ftp/arxiv/papers/1710/1710.11001.pdf, [Retrieved: April 2018].

[21] H.F. Atlam, R. J. Walters and G.B. Wills, "Fog Computing and the Internet of Things: A Review", Big Data Cogn. Comput. 2018, pp.2- 18; doi:10.3390/bdcc2020010, https://www.mdpi.com/2504-2289/2/2/10/pdf, [Retrieved: April 2018]

[22] "Fog Computing: Keystone of Industrial IoT and Industry 4.0", https://www.fogworldcongress.com/conference/agenda/fog-computing-keystone-industry-40, [Retrieved: April 2018]

[23] E.Bacarelli, P.G.Vinueza Naranjo, M.Scarpinti, M.Shojafar, and J.H.Abawaji,"Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study", IEEE Access, vol. 5, 2017, pp. 9882-9910.

[24] http://openedgecomputing.org, [Retrieved: April 2018]

[25] R. Schuster and P. Ramachandran, Open Edge Computing – From Vision to Reality - 2016 OPNFV Design Summit, Berlin, Germany, http://docplayer.net/32036666-Open-edge-computing-from-vision-to-reality.html, [Retrieved: April 2018].