Performance Modelling of Smart Cyber-Physical Systems

Work-In-Progress Paper

Tomas Bures
Charles University, Czech Republic
bures@d3s.mff.cuni.cz

Vladimir Matena
Charles University, Czech Republic
matena@d3s.mff.cuni.cz

Raffaela Mirandola
Politecnico di Milano, Italy
raffaela.mirandola@polimi.it

Lorenzo Pagliari
Gran Sasso Science Institute, Italy
lorenzo.pagliari@gssi.it

Catia Trubiani
Gran Sasso Science Institute, Italy
catia.trubiani@gssi.it

ABSTRACT

Context: the dynamic nature of complex Cyber-Physical Systems (CPS) introduces new research challenges since they need to smartly self-adapt to changing situations in their environment. This triggers the usage of methodologies that keep track of changes and raise alarms whether extra-functional requirements (e.g., safety, reliability, performance) are violated.
Objective: this paper investigates the usage of software performance engineering techniques as support to provide a model-based performance evaluation of smart CPS. The goal is to understand at which extent performance models, specifically Queueing Networks (QN), are suitable to represent these dynamic scenarios.
Method and Results: we evaluate the performance characteristics of a smart parking application where cars need to communicate with hot-spots to find an empty spot to park. Through QN we are able to efficiently derive performance predictions that are compared with long-run simulations, and the relative error of model-based analysis results is no larger than 10% when transient or congestion states are discarded.
Conclusion: the usage of performance models is promising in this domain and our goal is to experiment further performance models in other CPS case studies to assess their effectiveness.

CCS CONCEPTS

• **Software and its engineering → Software performance**;

KEYWORDS

Cyber-Physical Systems; Model-based Performance Modelling; Queueing Networks.

ACM Reference Format:
Tomas Bures, Vladimir Matena, Raffaela Mirandola, Lorenzo Pagliari, and Catia Trubiani. 2018. Performance Modelling of Smart Cyber-Physical Systems: Work-In-Progress Paper. In *Proceedings of ACM/SPEC International Conference on Performance Engineering (ICPE'18 Companion).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3185768.3186306

ICPE'18 Companion, April 9–13, 2018, Berlin, Germany

ACM ISBN ISBN 978-1-4503-5629-9/18/04…$15.00
https://doi.org/10.1145/3185768.3186306

1 INTRODUCTION

Cyber-Physical Systems (CPS) spread in a wide range of application areas like health-care and medicine, power management, electric smart grid and renewal energy, industry automation and manufacturing systems [13], air traffic control [12] and aircraft avionics systems [5, 12], automotive with smart car and intelligent road system with unmanned vehicle [1, 16]. Such broad set of application areas makes the task of designing CPS not trivial at all, in fact it is necessary to combine multiple components with different levels of abstraction, and it is fundamental to evaluate their extra-functional properties (e.g., safety, reliability, performance) [5, 7].

In literature several approaches recently emerged for the modelling of CPS (e.g., [3, 5, 9, 11, 17]). Most of them focuses on functional analysis ([5, 9, 17]) or formal verification ([3, 11]), but performance-related characteristics result to be almost neglected. In this paper we focus on the performance evaluation of CPS at design time since our goal is to anticipate performance flaws and avoid late fixes during the testing and after the implementation. This early study allows engineers to evaluate the performance characteristics of alternative system configurations.

In our previous work [14], we presented a case study to elicit the challenges for the performance engineering of CPS, and we identified several challenges requiring a deeper investigation. Recently, we also proposed a guided process namely *IMPACt* (mult*I* *M*odelling *P*erform*A*nce *C*ps), which supports architects to better understand the behavior of the system through model-based performance analysis results [15]. It consists of four main steps: (1) CPS specification, i.e., the system is modelled and performance requirements are defined; (2) models definition, i.e., performance models are built to evaluate the system characteristics; (3) models validation, i.e., models predictions are compared with (theoretical, simulated, actual) results; (4) results analysis, i.e., multiple settings are evaluated to select the system configuration that better fits with the stated requirements. In this paper we investigate one feasible implementation of the *models definition and validation* steps by using Queueing Networks (QN) [10] as performance models. The validation is performed by comparing QN-based prediction results with simulation values that are obtained with an ad-hoc framework.

The paper is organized as follows. Section 2 reports our motivating example, Section 3 shows the performance modelling and experimental results, and Section 4 concludes the paper providing a discussion on raised challenges and future research directions.

2 MOTIVATING EXAMPLE

In this section we describe our motivating smart CPS example that is taken from the automotive domain since such domain emphasizes dynamicity and adaptability to the changing environment.

2.1 Scenario

In detail, our motivating scenario deals with cars collectively searching for a free parking place in the streets of a city. The motivation to our work builds upon a real life problem, i.e., the lack of parking places in the city centers. Specifically, our goal is to improve the procedure to get information about free parking places in historic city centers. The advantage of modern cities is the availability of buildings that provide many parking places together with occupancy monitoring. On contrary, historic cities does not have such structured parking or it is very limited. Drivers needs to rely on few parking places scattered across the area while the availability of a particular place is unknown. This usually leads to cars slowly roaming around the area desperately trying to find a free spot as close to the destination as possible. There are obvious disadvantages of finding a parking place this way. At least these include increased fuel consumption, higher traffic density and wasted time.

The obvious solution to the problem is the introduction of a park place registry and mandatory registration of every car occupying a parking place. Using this approach the parking availability is easy to share among the drivers. The problem is an assumption that every spot needs to be monitored for car presence, or every single car needs to be connected in order to maintain the registry. As of today neither of these assumptions is easy to met. In order to lower the assumptions a collective park place scanning system is here introduced and analyzed. The system consists of the smart connected cars that have the ability to capture the image of the surrounding area.

The system works as follows. The cars occasionally report their position to the edge cloud server that is in charge of controlling the area, so that the edge cloud is aware of the general position of each car. Once there is a car intending to find a parking place, the server searches trough the list of cars that are able to stream photos of their surrounding areas to the edge cloud server. Out of all the cars in the simulation 94% were at last once required to provide the parking place scan. The server processes the images captured by these scanning cars in order to find free parking places and reports possible results back to the car that initiated the search. If everything goes well the car can continue at full speed towards the parking position reported by the server.

The system is assumed to use mobile broadband data connection in order to access edge cloud server located at the front of the mobile network. The network related properties of the system are very important as delays in the packet delivery and overall connectivity problems can result in significantly degraded performance of the system. The same is true for lack of edge cloud processing capacity. Both of these lead to increased latency between image capture and result delivery. A low latency is very important for smooth function of the system. It is very difficult for a car to make a u-turn in a heavy traffic or while passing through a one-way street. The system can avoid such complications by selecting a scanning car that is in front of the parking car so that the parking car does not have to make a significant change to its heading. Naturally, such strategy is only effective when the communication and processing latency is reasonably low or at last predictable.

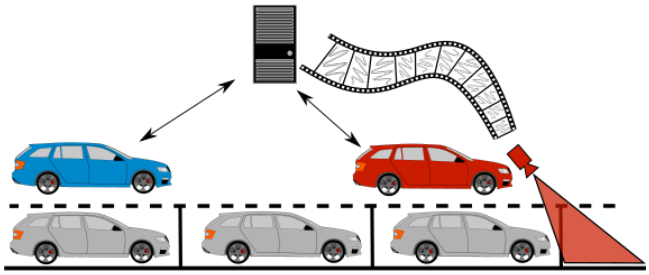

Figure 1: Motivating Example setup. Scanning cars (on the right) are sending a photo stream to the edge cloud server reporting spot availability to the parking cars (on the left).

Due to the requirement on the scan-to-response latency of the system the design time evaluation of system properties results to be very important. To this end, we pursue the application of IMPACt as a way to improve the whole system development. Indeed, without the ability to assess the system properties at design time it is much more difficult to implement a system that provides parking places on time and at the suitable location. These properties can be captured by measurements of the real system, or by precise simulation of both the system components and the mobile network. Both these approaches are of limited use. The real system measurements are lengthy and does not reveal the flaws at the design time. One possible solution is to adopt a simulation model in IMPACt application. However, the simulation takes very long time to obtain conclusive results thus it makes parameter tuning a time demanding process. These limitations lead to the investigation of different types of models for the system performance evaluation.

2.2 Simulation

The simulation of our motivating example has been conducted using the Veins LTE [8]. It is a framework based on the OMNeT++[1] simulator, combining radio simulation provided by the INET[2] and the SimuLTE[3] with vehicular simulation provided by the SUMO[4] framework. On top of the simulator libraries and modules two applications have been defined. The first one is called Parking Place Manager (PPM) and runs on the edge cloud server connected directly to the eNodeB mobile network cell. The second one is called Parking Place Scanning application (PPS) and runs on all cars involved in the scenario under analysis. The simulation has been set in a map of the Lesser town of Prague that has been extracted from the OpenStreetMap[5]. The simulation consists of a randomized stream of cars passing through the city district. The rate of the cars in the stream is controlled by the car emergence probability value. The simulation setup is visualized in Figure 2.

The PPM application receives status updates and scans data from cars. Received status updates are evaluated every unit of time in

[1] https://www.omnetpp.org
[2] https://inet.omnetpp.org
[3] http://simulte.com
[4] http://sumo.dlr.de
[5] http://www.openstreetmap.org

Figure 2: Simulation by OMNeT++ and veinsLTE.

order to decide which cars will be contacted to provide surroundings scans. The evaluation is based on the position and the status (i.e., parking or passing by) of the car. Position and heading of the car is used to choose a suitable scan provider for parking cars. The scan data received from cars are stored in a FIFO queue and processed by a simulated parking place analysis algorithm. Processing of a single scan is simulated as a 50ms delay.

The PPS application, which is running on each car, initially decides whenever the car is parking or just passing. This is a random decision. Each car is capable of providing a surroundings scan, even the cars that seek to find a parking place. The PPS application is using LTE network to send status updates to the edge cloud server hosting the PPM application. The status contains car mode (i.e., parking or passing by), current road id, heading, position, and assigned edge cloud server. When the car receives a scan request from the manager it initiates a scan and sends scan data twice per unit of time to the manager.

Initially the simulation was used to obtain timings of different actions in the system. These, together with configured periods are presented in Table 1. Based on the initial experiments it has been decided to run approximately 90 simulation runs different in the car emergence probability value ranging from 0.01 to 0.50. Moreover, it has been decided to simulate one hour of operation per each run in order to obtain stable results. As so many simulation runs would take very long to complete on a desktop computer, the simulation has been performed in parallel on a server. The simulations were executed on a server equipped with 128GiB of RAM and 2 Intel Xeon E5-2640v2 CPUs clocked at 2.00GHz. Due to excessive RAM usage the capability of the server to process 32 threads in parallel was not fully utilized. Thus, the executions of different runs did not interfere with each other. The average execution time of a single simulation run was roughly eleven thousands seconds [6].

3 PERFORMANCE MODELLING

As stated before the simulation time for the whole system is quite long, so to be able to quickly have some insights about the system performance, we investigate the use of Queueing Networks models in IMPACt. To this end, the QN model described in Figure 3 has been built for the car parking scenario. It basically reflects the behavior presented in Section 2, in particular cars send the status report to the server that contacts some cars to scan data and subsequently it scans both data and results. As QN product-form theory [2] suggests, clients (that in our case study are represented by different types of cars) and servers operations are modelled with queueing centers, whereas network connections are modelled as delay centers.

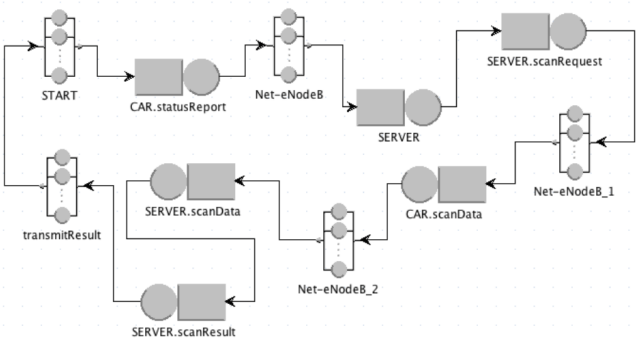

Figure 3: QN model for the car parking scenario.

As input parameters, we used the ones obtained by simulation and reported in Table 1. For example, the first row of Table 1 expresses that the status report from cars is provided with an average rate of 8.8 UT (unit of time), and this parameter is used by the QN solver to derive indicators on the system performance. A higher rate translates into a faster activity, in fact all entries showing one thousand as rate represent very fast activities, mostly delivered server side. The QN performance indices are then obtained by means of the Java Modeling Tool (JMT) [4].

Service Center	Input Parameters (UT)
CAR.statusReport	8.8
SERVER	1000
SERVER.scanRequest	4.8
CAR.scanData	1000
SERVER.scanData	100
SERVER.scanResult	1000

Table 1: QN input parameters.

Experimental results are reported in Table 2 where the first column reports the information on the considered number of cars. The last four columns respectively report: (i) the system response time expected with QN; (ii) the system response time simulated with an ad-hoc framework; (iii) the percentage error is calculated to estimated the gap between predicted and simulated values, both expressed in milliseconds (ms); (iv) the expected utilization of the server. For instance, in the first line we can see that the performance analysis is considering a parking scenario where one car provides the status report and two cars provide scan information (i.e., the total number of cars is three). The expected system response time is 93.89 milliseconds, the same evaluation but with the simulation

[6]To replicate the experiments, a repository containing all the source code and the scripts used to run the simulation is publicly available at: https://github.com/d3scomp/ParkingPlaceScanning.

framework provides a value of 119.43 milliseconds, hence in this specific setting the average relative error is 21.4%. However, we can additionally notice that the server shows a low utilization, i.e., 18%, denoting the system is warming up. Table 2 provides evidence that prediction errors are larger than 10% when the server shows low (i.e., lower than 50%) and high (i.e., higher than 80%) utilization, and these scenarios represent *"unstable"* states of warm-up and congestion, respectively. Shaded entries of Table 2 highlight *"stable"* system states where we get the most accurate predictions.

#cars	RT-prediction (ms)	RT-simulation (ms)	Error (%)	U-server (%)
3	93.89	119.43	21.4	18
7	137.08	114.25	19.98	43
10	143.67	135.72	5.8	61
13	187.09	170.89	9.4	75
16	249.67	190.92	30.8	88
20	343.48	449.31	23.5	97

Table 2: Experimental numerical results.

Figure 4 graphically shows the average number of cars on the x-axis, the average end-to-end latency (expressed in milliseconds) on the left-side of the y-axis, and the average queue length of the server on the right-side of the y-axis. This last information is helpful to identify congestion states where requests incoming to the server have to wait before being served. In fact, we can notice that the server queue length follows the same values estimated as latency up to 15 cars roughly, and then starts diverging a bit by denoting a certain system congestion.

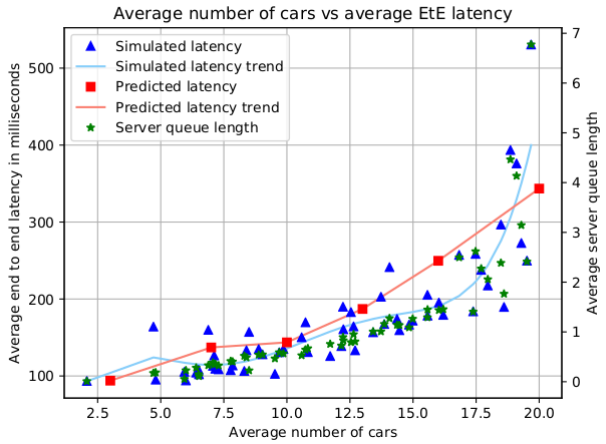

Figure 4: Experimental graphical results.

Summarizing, we can conclude that QN predictions are quite accurate when no system congestion is experienced in the system. This can be estimated by calculating the server queue length, in fact in our experimentation we found that the increasing of such value leads to a substantial gap between simulated and model-based predicted values. This last point of identifying the system congestion represents a crucial aspect that we want to investigate in the near future, and it also discussed in the next section where we collect all the identified open issues that require further investigation.

4 DISCUSSION AND CONCLUDING REMARKS

In this paper we have presented an exploratory study to investigate the models that can be used for the performance analysis of CPS during the design phase. To this end, we have followed the IMPACt approach and we have considered, for a car parking system, the effects of applying a simulation model and a more abstract QN model. The first results obtained with this exploratory study highlight some key aspects: (i) the ad hoc simulation models offer precise results but require a long time; (ii) the QN results cannot be used during all the system lifecycle because transient and congestion behaviors are not easily captured by these models; (iii) the need of more powerful models able both to capture the whole system behavior and to provide performance results in short time.

As future work, we intend to investigate real world scenarios where multi-modelling approaches would be able to deal with different aspects of complex systems. A promising set of models are the ones offered by the Ptolemy framework (http://ptolemy.eecs.berkeley.edu/index.htm). From a more general point of view, independently of the formalism adopted for the performance models, there is the need of an integrated approach that allows the automatic derivation of performance models starting from the definition of the system behavior. In this direction, we plan to apply DEECo (Dependable Emergent Ensembles of Components) that has been recently used for architecting software-intensive CPS [6].

REFERENCES

[1] Radhakisan Baheti and Helen Gill. 2011. Cyber-physical systems. *The impact of control technology* 12 (2011), 161–166.

[2] Simonetta Balsamo and Vittoria de Nitto Personè. 1994. A survey of product form queueing networks with blocking and their equivalences. *Annals of Operations research* 48, 1 (1994), 31–61.

[3] Steffen et Al. Becker. 2012. The MechatronicUML design method– process, syntax, and semantics. *SE group, University of Paderborn, Tech. Rep. tr-ri-12-326* (2012).

[4] Giuliano Casale and Giuseppe Serazzi. 2011. Quantitative system evaluation with Java modeling tools. In *WOSP/SIPEW Conference*. 449–454.

[5] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. 2012. Modeling cyber–physical systems. *IEEE* 100, 1 (2012), 13–28.

[6] Rima Al Ali et Al. 2014. DEECo: an ecosystem for cyber-physical systems. In *ICSE Companion*. 610–611.

[7] Allan Edgard Silva Freitas and Romildo Martins da Silva Bezerra. 2016. Performance Evaluation of Cyber-Physical Systems. *ICIC Express Letters* 10 (2016).

[8] Florian Hagenauer, Falko Dressler, and Christoph Sommer. 2014. Poster: A simulator for heterogeneous vehicular networks. In *2014 IEEE Vehicular Networking Conference (VNC)*. 185–186. https://doi.org/10.1109/VNC.2014.7013339

[9] Gabor Karsai and Janos Sztipanovits. 2008. Model-integrated development of cyber-physical systems. In *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*. Springer.

[10] L. Kleinrock. 1975. *Queueing Systems Vol. 1:Theory*. Wiley.

[11] Peter Gorm Larsen and et al. 2016. Towards semantically integrated models and tools for cyber-physical systems design. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 171–186.

[12] Edward A Lee. 2015. The past, present and future of cyber-physical systems: A focus on models. *Sensors* 15, 3 (2015), 4837–4869.

[13] Jay Lee, Behrad Bagheri, and Hung-An Kao. 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters* 3 (2015), 18–23.

[14] Lorenzo Pagliari, Raffaela Mirandola, and Catia Trubiani. 2017. A Case Study to Elicit Challenges for Performance Engineering of Cyber Physical Systems. In *ICPE Companion*. ACM, 217–222.

[15] Lorenzo Pagliari, Raffaela Mirandola, and Catia Trubiani. 2017. Multi-modeling approach to performance engineering of Cyber-Physical Systems design. In *International Conference on Engineering of Complex Computer Systems (ICECCS)*.

[16] Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. 2011. A survey of Cyber-Physical Systems. In *International Conference on Wireless Communications & Signal Processing WCSP*. 1–6.

[17] Muhammad Umer Tariq and et al. 2014. Design Specification of Cyber-Physical Systems: Towards a Domain-Specific Modeling Language based on Simulink, Eclipse Modeling Framework, and Giotto.. In *ACESMB@ MoDELS*. 6–15.