Sensyml: Simulation Environment for large-scale IoT Applications

Isakovic Haris*, Vanja Bisanovic†, Bernhard Wally*, Thomas Rausch*, Denise Ratasich*,
Schahram Dustdar*, Gerti Kappel*, Radu Grosu*
TU Wien, Vienna, Austria
* name.surname@tuwien.ac.at
† name.surname@student.tuwien.ac.at

***Abstract*—IoT systems are becoming an increasingly important component of the civil and industrial infrastructure. With the growth of these IoT ecosystems, their complexity is also growing exponentially. In this paper we explore the problem of testing and evaluating large scale IoT systems at design time. To this end we employ simulated sensors with the physical and geographical characteristics of real sensors. Moreover, we propose Sensyml, a simulation environment that is capable of generating big data from cyber-physical models and real-world data. To the best of our knowledge it is the first approach to use a hybrid integration of real and simulated sensor data, that is also capable of being integrated into existing IoT systems. Sensyml is a cloud based Infrastructure-as-a-Service (IaaS) system that enables users to test both functionality and scalability of their IoT applications.**

***Index Terms*—IoT, scalability, System-of-Systems, simulation, sensors**

I. Introduction

The Internet-of-things (IoT) is a multidisciplinary ecosystem under constant development and expansion. The concept emerged from a simple idea, namely to connect household devices or "things" with users or other things over the internet. For example *a sensor in an electrical outlet is measuring electrical current and provides a user with information whether an appliance connected to this outlet is active or not and what is the energy consumption of this appliance.* However, IoT systems are evolving from application oriented *"ad-hoc"* systems like this, towards hierarchically structured and highly integrated large-scale systems. It is projected that the number of IoT devices will reach 100 billion units until 2030 [1]. The increased number of IoT devices is proportional to the growth of data produced and stored annually. A report published in 2018 projected the growth of the global datasphere to astonishing 175 ZB in 2025 [2].

An IoT system consists of three major components: the swarm, the fog and the cloud. The swarm is akin to our skin and muscle cells. It consists of millions of sensors and actuators that are used to interact with the things. The fog is akin to our backbone. It is network of relatively simple computers, close to the swarm, and used for routing and real time control. Finally, the cloud is akin to our brain. It is a large computer farm used for data storage, planning and machine learning. The IoT hosts multiple applications that are designed and operated by different organizations. The applications have to operate in parallel and share existing infrastructure. Services created by each application must be communal within a system, to avoid unnecessary redundancy and reduce resource overheads. A Cyber-physical system combines digital representation of a system and physical description to observe behavior of the system [3]. A combination of CPS and IoT systems can be classified as System-of-System(SoS) [4]. As such system is able to adapt to emerging conditions, and learn about the physical system and its behavior. In general, it is quite difficult to predict in advance data throughput and the adequate dimension the IoT infrastructure precisely, and therefore simulation is a convenient way of exploring such properties. In this paper we are proposing a simulation environment called *Sensyml* that is capable of simulating large scale data during application development. Sensyml is capable of hosting multiple types of sensors with physical and geographical characteristics similar to real sensors, where sensors can be derived from mathematical models or extracted from existing online services. Moreover, Sensyml allows us to explore the capability of the IoT to satisfy its given specification under emerging conditions (e.g., power limitations, data growth). The simulation environment can be integrated in existing hardware infrastructure to test basic functions of the IoT and how the IoT scales up.

The rest of the paper is organized as follows. Section II provides the technical and theoretical background. Section III describes state-of-the-art. Implementation details for Sensyml are described in Section IV. Section V evaluates Sensyml and reflects on its application of the in real-world use cases. Finally in last two sections we discuss future work and provide final remarks.

Parts of this work have been supported by funding from the Austrian Government through the Federal Ministry Of Education, Science And Research (BMWFW) in the program Hochschulraum-Strukturmittel 2016 (HRSM).

II. Background

A. CPS/IoT Ecosystem Infrastructure

The project *CPS/IoT Ecosystem* [5] [6] explores synergies between cyber-physical systems (CPS) and IoT as a cornerstone infrastructure for development of smart services and applications. The IoT is evolving from singular application oriented systems towards heterogeneous clusters of applications sharing underlying hardware and/or software.

a) IoT as an Infrastructure: A generalized definition of an infrastructure states: "*An infrastructure is the basic physical and organizational structures and facilities (e.g. buildings, roads, power supplies) needed for the operation of a society or enterprise* [7]". Therefore, a system that supports structures of higher level of operation or functionality is part of an infrastructure. The project CPS/IoT Ecosystem organizes IoT in three scopes of operation: *cloud*, *fog*, and *sensor/actuator*. Each scope is represented by its functionality spectrum and its ability to perform certain tasks.

b) Cloud: Cloud systems are built on high-performance computation devices capable of handling large amounts of data (alias Big Data). They are commonly located of-site in large data centers and reachable exclusively through Internet. Sensors/actuators are low performance devices that interact with the physical environment.

c) Sensor/Actuator: A sensor measures a certain physical property and this data can be used to provide novel aspects for aware applications, to optimize a process or improve user experience. In Section I we described how an electrical current sensor is increasing the functional spectrum of a simple electrical outlet.

d) Fog: The effort of communicating data between these two layers is often underestimated. The communication and computational devices between the sensors and the cloud we refer as Fog. They allow mid-range performance and wide spectrum of communication standards. Fog devices provide the ability to handle, aggregate, and filter data relatively close to the source and with relatively small latency.

B. Everything-as-a-Service (XaaS)

A model where assets like infrastructure, hardware, software, intellectual property, testing or security are offered as a service is referred to as "as-a-Service" model [8]. Everything-as-a-Service (XaaS) is a term that comprises all the different models that use the "aaS" concept. Infrastructure-as-a-Service (IaaS) is a concept introduced in cloud computing where users can lease computing resources over a specific time and cost model [9]. In CPS/IoT Ecosystem we are translating this model further to fog and sensor layers, building a heterogeneous platform where individual software and hardware components can be used as services. To estimate the growth factor of an application and ensure its scalability the CPS/IoT Ecosystem platform provides Testing-as-a-Service (TaaS) that includes simulation, evaluation and experimentation.

C. Scalability

Scalability was always a vague term because it is highly dependent on the domain and application where it is applied. The notion of scalable systems originated with hardware architectures and denoted ability of the system to maintain work efficiency for arbitrary number of processors and a variable algorithm size [10]. A scalable system is able to accommodate increasing number of objects and elements, while maintaining a graceful functionality, and being responsive to changes in size [11]. The scalability is commonly associated with performance properties, however scalability observed in other properties especially in distributed and software systems [12]. In this section we will to explore important aspects of scalability in respect to IoT as an infrastructure as defined in project CPS/IoT Ecosystem [13] [6].

According to Bondi in [11] scalability can be classified in four groups: load scalability, space scalability, space-time scalability and structural scalability. In distributed system scalability can be manifested in distance or speed in relation to distance. *Load scalability* represents the ability of the system to continue to function with moderate and tolerable decline in performance under increased workload. The system should behave in a similar way under low, moderate and high work load. *Space scalability* is related to memory consumption of a system. If the number of tasks increases memory consumption should remain within reasonable limits. *Space-time scalability* refers to the ability of a system to function seamlessly or with reasonable decline in performance even if the number of tasks increases by an order of magnitude. *Structural scalability* states that an implementation of a system and standards and regulation allow the system to expand, increasing the number tasks it can perform or number of objects it can serve, without additional changes to the implementation or standards. This can hold either indefinitely or for a specific time frame. Systems could be allowed to scale in one aspect but they could be limited in other, giving the system ability to grow only within certain period of operation.

D. Modeling and Simulation

Having formal models of CPS/IoT systems at hand opens up powerful capabilities with respect to planning, simulation, programming, deployment and maintenance [14]. CPS/IoT systems cover an extremely wide range of environments that need to be considered in order to gain meaningful insights into their workings, including views on, e.g., embedded systems, network topologies and information flow.

An important aspect of well-defined models is that they can be used for the creation of executable artifacts (code generation) [15]. Depending on the level of detail provided in the models and depending on the context of the code generation, one can imagine different kinds of generated artifacts, including high-level configuration information or compilable ANSI C code [16]. In the context of CPSs, it might be tempting to generate embedded code based on the hardware configuration of the embedded system (e.g., changing the port assignment of a sensor would be reflected in the code generation by injecting the correct pin number into the code). In the context of IoT, a change in the network topology could lead to changes in the code of embedded systems with respect to the internet protocol (IP) addresses or hosts an embedded system would connect to, or the software communication protocol or payload format could be switched from one to another and thus the respective parts of the code could be "bulk"-replaced.

When it is possible to generate code fragments or even complete systems from formal models, this code generation can be triggered by higher-order code generation frameworks and produce multiple variants of code with distinct variations. Thus, it enables the creation of numerous "cyber" entities, which leads to the ability to simulate the behavior of a CPS system without the need to physically prepare and deploy embedded systems. These simulation entities could be generated with behavior that follows a certain function and/or they could employ some random behavior, depending on the configuration of the higher-order code generation framework.

The application of model-driven engineering techniques to the field of CPS/IoT application development has been investigated, e.g., in [17]. While this study did not explicitly consider "simulation" as a target application, it shows that the main focus of models in the IoT domain was on creating semantic or meta models with the aim of describing deployment issues, service composition, devices themselves, and required middleware. We believe, that with these pieces of information, it is feasible to generate executable code fragments that can be used for simulation efforts.

To reasonably evaluate the scalabiliy of a CPS/IoT infrastructure or application it is required to be able to set the load on processing units, communication channels, storage devices, power supplies to reasonable limits and observe the behavior of the system under these conditions. Leveraging models can lead to the generation of many CPS/IoT entities in a plethora of configurations and deployed on various computing nodes.

E. Quality of Service in IoT

Many modern IoT scenarios have stringent quality of service (QoS) requirements that cannot be met by cloud-based solutions alone. In particular in latency-sensitive IoT scenarios, message routing to the cloud incurs round-trip times that can be detrimental to the application. Low-latency communication and data processing near the edge is therefore a critical requirement for satisfying these QoS requirements [18].A fundamental issue that confronts Fog computing systems in IoT scenarios is proximity detection between sensors, actuators, and edge resources; and a key enabler for proximity detection is network QoS monitoring. Furthermore, to enact service level agreements (SLAs) for IoT scenarios, a system requires detailed and timely information about network QoS. Research has shown that straight-forward solutions to network QoS monitoring do not scale well with increasing number of network nodes [19], and that established approaches in cloud-computing, e.g., proximity detection in CDNs, have serious limitations when applied in IoT scenarios [20]. Specifically, high-resolution QoS monitoring, necessary for both proximity detection in mobile scenarios, as well as SLA enactment, can put a high strain on the network. IoT scenarios further exacerbate this problem, due to the immense amount of networked devices involved.

III. Related Work

Simulation environments can be organized in several groups: full stack simulators, big data processing, network simulators [21]. From the perspective of scalability all three groups are important in their own scope, depending what aspect of the system we want to evaluate. In this work we are focusing on the scalability of the infrastructure and data processing. This considers load scalability of the fog nodes and cloud virtual machines due to increased number of sensors. We identified number of simulators that could be used to evaluate load scalability e.g., iFogSim [22], DPWSim [23], CloudSim [24], IoTSim [25], SimIoT [26]. Simulation environments can approximate production conditions of a system to a certain level. They depend on the accuracy of the models and resources they are allowed to use. In order to be able to validate results experimentation and testing environments need to be reproducible and repeatable. To achieve this research and commercial communities introduced concept of an IoT testbed where users can implement their applications and perform evaluation on existing infrastructure. Few research initiatives created mid and large-scale IoT testbeds to bring smart city and IoT infrastructure closer to developers without the need to build costly individual experimental setups. Fit/IoT LAB [27] (initially known as SensLAB) is a testbed project in France with over 1500 wireless sensor nodes. It is a multi-user platform for research on wireless nodes with standard and custom node possibilities, also it provides a mobility research platform based on mobile robots. Other testbeds include SmartSantander [28] an European project with goal to build large scale IoT network in four European cities. Japanese project JOSE [29] short for "Japan-wide orchestrated smart/sensor environment" also provides the ability to virtualize infrastructure and execute multiple applications and IoT services at the same time. The testbeds provide ability to validate application designs on large scale networks. However, they are still bound with a fixed number of sensors. Combining functionality of a testbed with a simulation environment like Sensyml provides more realistic evaluation scenarios with the ability to scale number of sensors or fog devices.

Scaling an application from a prototype to large distributed systems in IoT: the heterogeneous nature of IoT makes it almost impossible to have a single simulation environment for the whole application. Existing IoT simulation frameworks focus on exploring design properties of the IoT system by establishing typologies and related constraints. Sensyml focuses on big data generation using sensors created from mathematical models or real existing data (Internet, database). We are able to simulate multiple heterogeneous sensors and connect them to existing infrastructure (fog and cloud nodes) to test its functionality or scalability.

Testbeds focus on real life examples with specific use cases where they create an arbitrary number of nodes. In order to see full potential of such application it is necessary to implement large numbers of sensors. It is difficult to fully evaluate the capabilities of fog or cloud software and hardware. Sensyml is a simulation platforms that augments any IoT testbed by allowing it to model any type of sensor and simulate almost any number of these sensors. The simulation environment is dynamic and cloud based utilizing micro-services technology to ensure scalability of the simulation system itself. It provides interface virtualization such that fog and cloud applications can use existing interfaces to connect to individual sensors or group of sensors. Furthermore, it can be combined with heterogeneous infrastructures and can be accessed by multiple users at the same time. Section IV provides a description of the Sensyml simulation environment and its practical application.

IV. Sensyml: Simulator for IoT Sensor Swarm

In this section we present a cloud based simulation environment for IoT sensor nodes. It provides an extension to the CPS/IoT Ecosystem and allows users to spawn and simulate various sensors with geographical and physical relation.

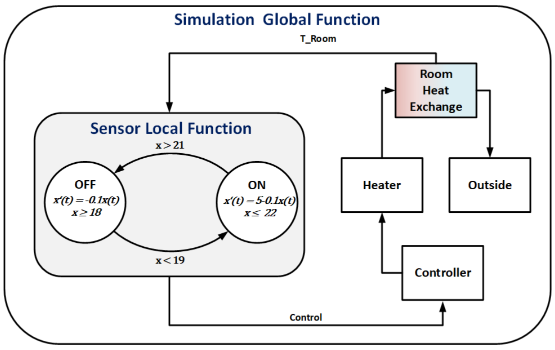


Fig. 1. Room Temperature Simulation Model.

A. Simulation Environment

Sensmyl uses software agents to implement simulated sensor functions within a larger simulation environment. Sensor functions can be derivatives of cyber-physical models (see Figure 1) from tools like MATLAB [30], simple models with pseudo-random generated parameters, or model with randomized values extracted from historic data of a physical sensors, or real-world data services (e.g., OpenWeatherMap [31]). The global simulation environment function provides input to the sensors. Sensyml provides interface virtualization that allows users to connect to the individual sensors. The connection from the user side to the sensors is seamless and requires no additional software from the user side. This allows us to abstract from the hardware platforms and still be compatible with existing network and software interfaces. Sensyml can be connected to the actual infrastructure e.g., CPS/IoT Ecosystem or virtualized infrastructure implemented in cloud. Figure 2 provides typical structure of an CPS/IoT system with Sensyml.

B. Technology Overview

The Sensyml simulation environment has been developed in Java 8 [32] as a cloud application with web based user interface. This means, that the system can run and scale at any modern Java EE Server [33]. Currently, the Sensyml system is running on a Wildfly Application Server [34] version 14. Other tools used in the development process are Maven [35] for the build management and Mongo DB [36], an open source NoSQL database. This development pipeline allows Sensyml to be platform independent and can be applied both in cloud

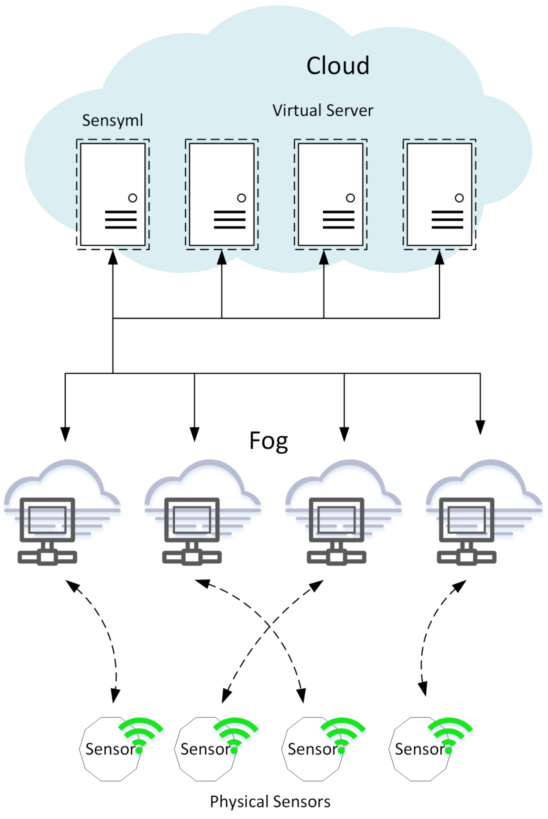


Fig. 2. IoT Testbed Structure with Sensyml.

and edge/fog layer. In current version Sensyml interfacing mechanisms are based on HTTP and HTTPS protocols.

C. Software Architecture

Sensyml consists of six basic software modules depicted (see Figure 3): *sensyml-model*, *sensyml-database*, *sensyml-creator*, *sensyml-control*, *sensyml-web* and *sensyml-interfaces*. The *sensyml-model* is central component in the architecture and all other modules are directly or indirectly dependent on it. It contains the sensor object templates which are used to generate specific sensors or sensor networks. The *sensyml-database* module contains all the database objects and repositories used for direct operations over the database The *sensyml-creator* module is used to generate different types of sensors as single sensors or in batches. It is directly interfaced with the *sensyml-database* module, what makes it possible to add new sensors independent of the simulation environment. The *sensyml-control* module is responsible for the initialization and maintenance of the ongoing simulation, and also provide service layer for Web and Interface modules. The *sensyml-web* module is a user interface for the simulation environment. It allows users to interact with the simulation or individual sensors. The *sensyml-interface* module provides provides sensors interface virtualization allowing users to connect clients to specific sensors or sensors groups.

All specific sensor services are abstracted from a generic sensor service. Figure 4 shows a class diagram that provides

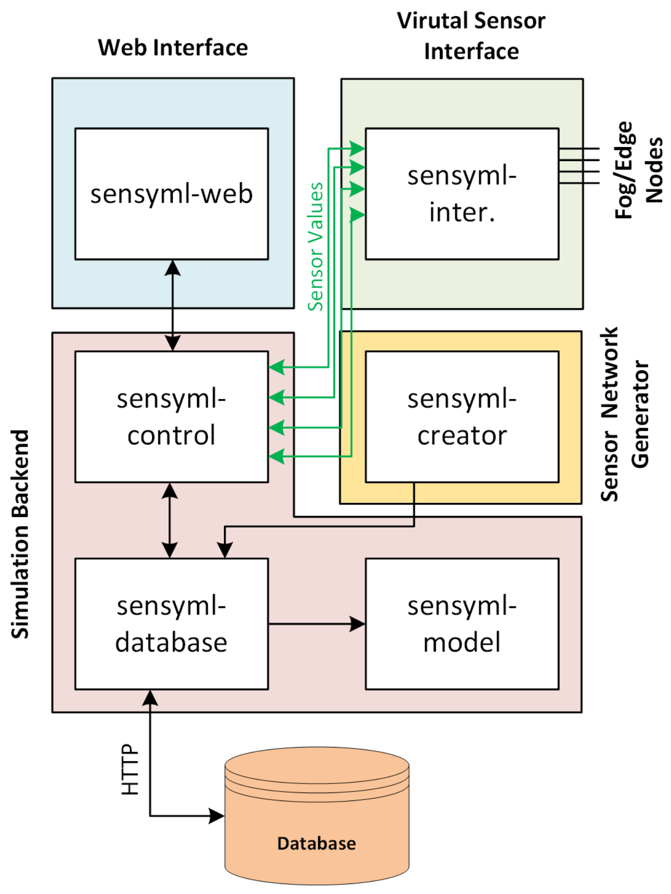


Fig. 3. Sensyml Software Architecture.

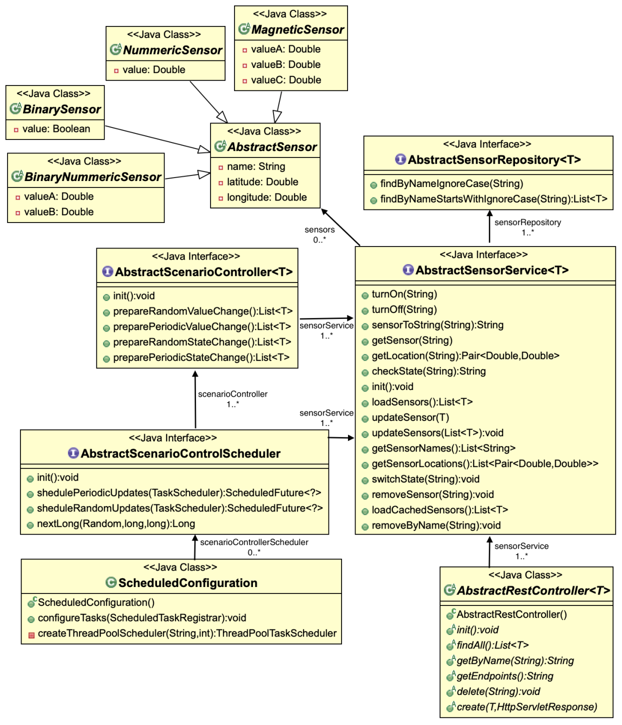

<<Java Class>> NummericSensor
value: Double
<<Java Class>> MagneticSensor
valueA: Double
valueB: Double
valueC: Double
<<Java Class>> BinarySensor
value: Boolean
<<Java Class>> BinaryNummericSensor
valueA: Double
valueB: Double
<<Java Class>> AbstractSensor
name: String
latitude: Double
longitude: Double
<<Java Interface>> AbstractSensorRepository<T>
findByNameIgnoreCase(String)
findByNameStartsWithIgnoreCase(String):List<T>
sensors 0..*
sensorRepository 1..*
<<Java Interface>> AbstractScenarioController<T>
init():void
prepareRandomValueChange():List<T>
preparePeriodicValueChange():List<T>
prepareRandomStateChange():List<T>
preparePeriodicStateChange():List<T>
<<Java Interface>> AbstractSensorService<T>
turnOn(String)
turnOff(String)
sensorToString(String):String
getSensor(String)
getLocation(String):Pair<Double,Double>
checkState(String):String
init():void
loadSensors():List<T>
updateSensor(T)
updateSensors(List<T>):void
getSensorNames():List<String>
getSensorLocations():List<Pair<Double,Double>>
switchState(String):void
removeSensor(String):void
loadCachedSensors():List<T>
removeByName(String):void
sensorService 1..*
scenarioController 1..*
sensorService 1..*
<<Java Interface>> AbstractScenarioControlScheduler
init():void
shedulePeriodicUpdates(TaskScheduler):ScheduledFuture<?>
sheduleRandomUpdates(TaskScheduler):ScheduledFuture<?>
nextLong(Random,long,long):Long
scenarioControllerScheduler 0..*
sensorService 1..*
<<Java Class>> ScheduledConfiguration
ScheduledConfiguration()
configureTasks(ScheduledTaskRegistrar):void
createThreadPoolScheduler(String,int):ThreadPoolTaskScheduler
<<Java Class>> AbstractRestController<T>
AbstractRestController()
init():void
findAll():List<T>
getByName(String):String
getEndpoints():String
delete(String):void
create(T,HttpServletResponse)

Fig. 4. Sensyml Class Diagram - Abstract Classes.

an insight in relationship between individual sensor simulation and global environment simulation. In Sensyml virtual sensor nodes can be created in three ways: a) Sensylm-creator batch application b) In-browser management application where the sensors can be created manually as simple as placing them on a map, c) Sensyml-interface web application, by using REST and WS endpoints clients can connect and remotely manage the data over HTTP/HTTPS protocol. This allows highly flexible management of the simulation environment and interoperability with high variety of clients. Internal state of each sensor is recorded in the database. The simulator configuration can be defined in advance or during runtime. It is capable of running multiple simulation scenarios simultaneously. It is also possible to log sensor values over time and observe historical changes. Such data is valuable because it could be used for offline analysis, to improve models and optimize simulation. The system is designed to be able to connect with a single database or a cluster in a completely transparent way. This feature ensures better scalability of the system, especially for large scale simulations.

V. Results and Discussion

The main objective of Sensyml is to generate sensor data on a large scale and the ability to connect these individual data sources to existing IoT infrastructure. The data can be used to evaluate scalability, to perform functional tests, as a learning data for algorithms. Further, this platform should be scalable and provide generic interfaces towards existing infrastructure components. Under infrastructure we consider cloud and fog/edge devices that are either part of an existing testbed (e.g., CPS/IoT Ecosystem) or virtualized IoT infrastructure implemented in cloud. At current stage we tested the simulation environment using Smart City use case with up to 50000 sensors. Further we successfully ran the simulation with multiple types of sensors and simulation functions. Obviously the performance of the simulator is effected by the amount of computing resources available. However, we need to ensure that the functionality of the simulation framework remains reasonably intact under increased load. Table I shows round trip times measured from a client machine for sensor requests, with two different hardware setups on the server sides and different number of sensors per request. We notice that RTT is only slightly affected by reduced hardware performance. Moreover, with increase of sensors by 10-fold the RTT increases by in average only by 2-fold. Table II provides preliminary performance measurements for the same experiment. The request run is performed in four stages 10, 100, 1000, 1000 sensors per request, and for each group we perform 1000 requests (see Request Run column). We observe that the memory and disk consumption are relatively low and remain constant with slight deviations in memory consumption. The CPU load is 27% higher in average during the experiment than the idle run, with the maximum load up to

95%. This scenario was performed on a static setup to evaluate overhead of the simulation framework. The performance is affected by different simulation functions etc. The platform is still in an early alpha version, and more elaborate test will be performed as the platform reaches production grade version. Further we present a Smart City use case and how can Sensyml be applied to design IoT infrastructure for a smart city.

TABLE I
ROUND TRIP TIME (RTT) MEASUREMENTS FOR A SENSOR READING REQUEST, BASED ON 1000 SAMPLES.

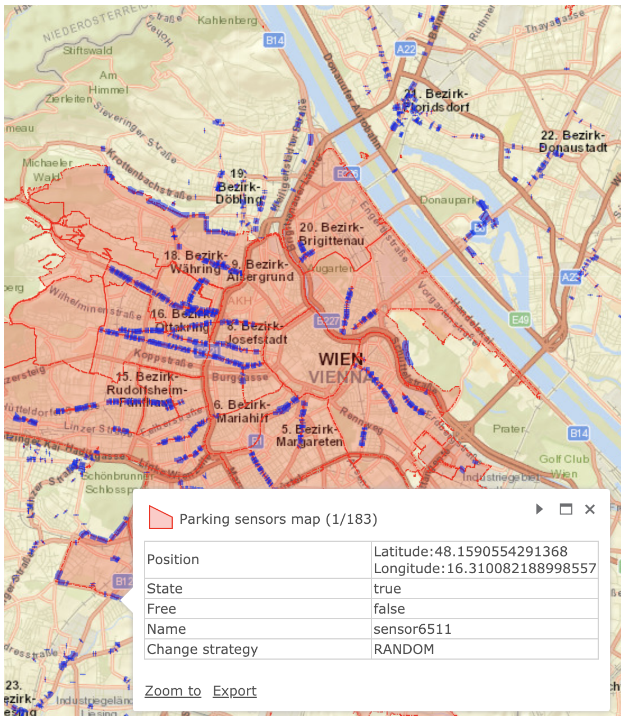
# Sens./Request	VM: 1 CPU Core, 4 GB		VM: 2 CPU Cores, 8 GB	
	Mean RTT (s)	Std. Dev (s)	Mean RTT (s)	Std. Dev (s)
10	0.058	0.010	0.056	0.002
100	0.083	0.011	0.083	0.033
1000	0.396	0.053	0.384	0.032
10000	0.634	0.132	0.610	0.142

TABLE II
SENSYML RESOURCE CONSUMPTION MEASUREMENT FOR SENSOR READ REQUEST. PERFORMED ON A VIRTUAL MACHINE WITH 2 CORES AND 8 GB OF RAM.

	Request Run			Idle		
	Mean (%)	Std. Dev	Max (%)	Mean (%)	Std. Dev	Max (%)
RAM	23.03	0.881	23.93	21.07	0,0606	21.25
CPU	58	0.168	95	40	0,12	67
DISK	36	0	36	36	0	35

a) Smart City Simulation Use Case: In this scenario Sensyml is used to evaluate smart city applications such as smart parking or environmental tracking on design time. These applications require large number of geographically distributed sensor nodes and required infrastructure is extremely difficult to test without actual physical nodes. On the other hand the implementation based on small scale prototypes poses a risk of underestimating the software and hardware resource requirements.

We assume the architecture of a smart city application to be based on simple sensor nodes that depend on fog/edge node gateways to deliver their data to the internet. For example, a street will have a number of sensor nodes and a single fog gateway to filter, aggregate and forward sensor data to the cloud. We assume this scenario with respect to the goal of sustainable IoT. This means sensor nodes are design to be reliable, energy independent and economical. If we consider only short parking zones in a city of Vienna the number of sensors adds up to 55372 sensors. This number can be even higher if we want to implement redundancy and higher precision. Sensyml provides smart city sensor simulation for two applications smart parking and environmental sensors. Smart parking sensor is based on a model of magnetic field sensor, and the climate sensors are based on randomized values from an online weather service OpenWeatherMap [31]. Both scenarios are using actual public geographical information about parking areas, zones and lanes, and locations of meteorological stations in the city of Vienna, Austria. First scenario concentrates on simulation of different in-advance prepared behaviors of parking sensor grids over real time.



Parking sensors map (1/183)

Position	Latitude:48.1590554291368 Longitude:16.310082188998557
State	true
Free	false
Name	sensor6511
Change strategy	RANDOM

Fig. 5. Smart-Parking simulation overview in Sensyml web application.

Second scenario additionally provides means to interfacing with the other already existing web service endpoints, such as Weather API. As such, it provides means of feeding real-time data into simulated sensors within Sensyml. Figure 5 shows simulation framework interface with geographical and sensor data.

VI. Future Work

Future work considers building a sensor library, such that prototyping can be (i) faster and (ii) more accessible for a wider user base. It also involves generalizing the framework to involve other simulation scenarios. Further, the framework should be extended in order to include support for a wider range of IoT protocols. One of the main goals in the future is to integrate Sensyml with existing IoT frameworks (e.g., Arrowhead [37]). This would ensure faster and more flexible prototyping of large scale applications. Further testing is key of the near future development as well as deployment on a publicly reachable platform such that it can be used by students and researchers.

VII. Conclusion

In this paper we have presented basic challenges in designing and implementing large-scale IoT applications. We conclude that scalability properties are especially difficult to comprehend in large heterogeneous IoT networks. The amount of data generated changes with each sensor, and analysis algorithms are getting more “hungry” by the day. The potentially massive amount of hardware and software components that lie in between need to be able cope with this trend. We have proposed a simulation environment that is able to simulate a large number of IoT sensor nodes and create data from models and real-world sources. It is deployed in a fully scalable environment, and able to blend with existing heterogeneous infrastructure.

References

[1] K. Rose, S. Eldridge, and L. Chapin, "The Internet of Things: An Overview," Feb. 2015, bibtex[howpublished=ISOC-IoT-Overview-20151221-en.pdf]. [Online]. Available: https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf

[2] D. Reinsel, J. Gantz, and J. Rydning, "The Digitization of the World from Edge to Core," p. 28, 2018.

[3] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed. The MIT Press, 2016.

[4] A. Ceccarelli, A. Bondavalli, B. Froemel, O. Hoeftberger, and H. Kopetz, "Basic Concepts on Systems of Systems," in *Cyber-Physical Systems of Systems: Foundations – A Conceptual Model and Some Derivations: The AMADEOS Legacy*, ser. Lecture Notes in Computer Science, A. Bondavalli, S. Bouchenak, and H. Kopetz, Eds. Cham: Springer International Publishing, 2016, pp. 1–39. [Online]. Available: https://doi.org/10.1007/978-3-319-47590-5_1

[5] "CPS/IoT Ecosystem: A platform for research and education – CPS/IoT Ecosystem." [Online]. Available: http://cpsiot.at/?p=193

[6] H. Isakovic, D. Ratasich, C. Hirsch, M. Platzer, B. Wally, T. Rausch, D. Nickovic, W. Krenn, S. Dustdar, and R. Grosu, "CPS/IoT Ecosystem: A platform for research and education," p. 8.

[7] "infrastructure | Definition of infrastructure in English by Oxford Dictionaries." [Online]. Available: https://en.oxforddictionaries.com/definition/infrastructure

[8] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu, "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends," in *2015 IEEE 8th International Conference on Cloud Computing*, Jun. 2015, pp. 621–628.

[9] A. Iosup, R. Prodan, and D. Epema, "IaaS Cloud Benchmarking: Approaches, Challenges, and Experience," in *Cloud Computing for Data-Intensive Applications*, X. Li and J. Qiu, Eds. New York, NY: Springer New York, 2014, pp. 83–104. [Online]. Available: https://doi.org/10.1007/978-1-4939-1905-5_4

[10] M. D. Hill, "What is Scalability?" *SIGARCH Comput. Archit. News*, vol. 18, no. 4, pp. 18–21, Dec. 1990. [Online]. Available: http://doi.acm.org/10.1145/121973.121975

[11] A. B. Bondi, "Characteristics of Scalability and Their Impact on Performance," in *Proceedings of the 2Nd International Workshop on Software and Performance*, ser. WOSP '00. New York, NY, USA: ACM, 2000, pp. 195–203. [Online]. Available: http://doi.acm.org/10.1145/350391.350432

[12] L. Duboc, D. Rosenblum, and T. Wicks, "A Framework for Characterization and Analysis of Software System Scalability," in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC-FSE '07. New York, NY, USA: ACM, 2007, pp. 375–384. [Online]. Available: http://doi.acm.org/10.1145/1287624.1287679

[13] "CPS/IoT Ecosystem – HRSM Project." [Online]. Available: http://cpsiot.at/

[14] J. Bézivin, "On the unification power of models," *Software and System Modeling*, vol. 4, pp. 171–188, 2005.

[15] S. Kent, "Model driven engineering," in *Integrated Formal Methods*, M. Butler, L. Petre, and K. Sere, Eds., 2002, pp. 286–298.

[16] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, 2nd ed., ser. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017.

[17] S. Wolny, A. Mazak, and B. Wally, "An initial mapping study on MDE4IoT," in *Proceedings of the 2nd International Workshop on Model-Driven Engineering for the Internet-of-Things (MDE4IoT)*, 2018.

[18] H. Truong and S. Dustdar, "Principles for Engineering IoT Cloud Systems," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68–76, Mar. 2015.

[19] T. Rausch, S. Nastic, and S. Dustdar, "EMMA: Distributed QoS-Aware MQTT Middleware for Edge Computing Applications," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, Apr. 2018, pp. 191–197.

[20] T. Rausch, S. Dustdar, and R. Ranjan, "Osmotic Message-Oriented Middleware for the Internet of Things," *IEEE Cloud Computing*, vol. 5, no. 2, pp. 17–25, Mar. 2018.

[21] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1637–1647, Jun. 2018.

[22] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments," *arXiv:1606.02007 [cs]*, Jun. 2016, arXiv: 1606.02007. [Online]. Available: http://arxiv.org/abs/1606.02007

[23] S. N. Han, G. M. Lee, N. Crespi, K. Heo, N. V. Luong, M. Brut, and P. Gatellier, "DPWSim: A simulation toolkit for IoT applications using devices profile for web services," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, Mar. 2014, pp. 544–547.

[24] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities," *arXiv:0907.4878 [cs]*, Jul. 2009, arXiv: 0907.4878. [Online]. Available: http://arxiv.org/abs/0907.4878

[25] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, "IOTSim: A simulator for analysing IoT applications," *Journal of Systems Architecture*, vol. 72, pp. 93–107, Jan. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762116300662

[26] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, "Towards Simulating the Internet of Things," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, May 2014, pp. 444–448.

[27] C. Burin des Roziers, G. Chelius, T. Ducrocq, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, T. Noél, and J. Vandaele, "Using SensLAB as a First Class Scientific Tool for Large Scale Wireless Sensor Network Experiments," in *NETWORKING 2011*, ser. Lecture Notes in Computer Science, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds. Springer Berlin Heidelberg, 2011, pp. 147–159.

[28] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer, "SmartSantander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, Mar. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004337

[29] "JOSE: An open testbed for field trials of large-scale IoT services." [Online]. Available: https://www.researchgate.net/publication/306143509_JOSE_An_open_testbed_for_field_trials_of_large-scale_IoT_services

[30] "MATLAB - MathWorks - MATLAB & Simulink." [Online]. Available: https://de.mathworks.com/products/matlab.html

[31] "Weather API - OpenWeatherMap." [Online]. Available: https://openweathermap.org/api

[32] "Java 8 Central." [Online]. Available: https://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html

[33] "Java EE Servers - Your First Cup: An Introduction to the Java EE Platform." [Online]. Available: https://docs.oracle.com/javaee/6/firstcup/doc/gcrkq.html

[34] "WildFly Homepage · WildFly." [Online]. Available: https://www.wildfly.org/

[35] "Maven – Introduction." [Online]. Available: https://maven.apache.org/what-is-maven.html

[36] "Open Source Document Database." [Online]. Available: https://www.mongodb.com/index

[37] J. Delsing, *IoT Automation: Arrowhead Framework*. CRC Press, 2017. [Online]. Available: https://books.google.at/books?id=6mMlDgAAQBAJ