Testing IoT systems using a hybrid simulation based testing approach

Noname manuscript No.
(will be inserted by the editor)

Testing IoT systems using a hybrid simulation based testing approach

Stig Bosmans · Siegfried Mercelis · Joachim Denil · Peter Hellinckx

Received: date / Accepted: date

Abstract This paper presents an extensive overview of the challenges that arise when testing large IoT applications at the system level. In order do that we start from analyzing behavior of local entities such as IoT devices or people interacting with the IoT system. The interactions of these local entities eventually leads to an emergent behavior. Both the emergent behavior and the local behavior need to be taken into account when testing IoT systems. Therefore, we present a novel hybrid simulation based testing approach that is able to effectively facilitate interactions of these local entities. Furthermore, we introduce various solutions to the challenges that arise when implementing this hybrid methodology. These challenges are mainly related to the IoT development pipeline, synchronization between real-life and simulation environment and the scalability constraints of modern simulation techniques.

Stig Bosmans
University of Antwerp
imec - IDLab
Groenenborgerlaan 171 - Antwerp
E-mail: stig.bosmans@uantwerpen.be

Siegfried Mercelis
University of Antwerp
imec - IDLab
Groenenborgerlaan 171 - Antwerp
E-mail: siegfried.mercelis@uantwerpen.be

Joachim Denil
University of Antwerp
Flanders Make
Groenenborgerlaan 171 - Antwerp
E-mail: joachim.denil@uantwerpen.be

Peter Hellinckx
University of Antwerp
imec - IDLab
Groenenborgerlaan 171 - Antwerp
E-mail: peter.hellinkcx@uantwerpen.be

Keywords Internet of Things · Participatory Sensing · Testing · Simulation · Emergent Behavior

1 Introduction

Internet of things is getting more mature. IoT related projects and methodologies have been described extensively in literature in the past few years. Complex IoT systems are rolled out in urban environments. All with the aim to improve public or private services such as Industry 4.0. The promise of IoT and the impact it will have on societies can not be underestimated.
In this paper we look at the challenges posed by testing large scale Internet of Things environments. The implicit heterogeneity of large scale IoT systems, which typically consist of many thousands of interacting actuators, sensors and people make this an incredibly challenging task. We notice that testing of Internet of Things systems at the system level has been largely ignored in state of the art. Based on the experience obtained by the development of a novel participatory sensing framework we provide an overview of the challenges that arise when testing large IoT systems at the system level.

A very important aspect when testing Internet of Things systems at the system level is to include the behavior exhibited by local entities (LE's). In order to orchestrate such LE behavior, the LE will need to be able to interact in real-time with the environment and with other entities. This eventually leads to a global, emergent behavior. Emergent behavior is the overall, global behavior posed by a system. It inherently depends on the behavior and interactions of LE's with each other and with the environment. Mataric et al. define emergent behavior as a collection of actions and patterns that result from local interactions between elements and their environment, which have not been explicitly programmed. When tested and calibrated correctly this emergent behavior can lead to various global optimizations at the system level. However, a large amount of real-time interacting LE's, posing realistic behavior will be required to properly test this.

We propose a hybrid simulation based testing technique to do this. On one end we rely on simulation of the behavior of LE's which has as an advantage that it is highly scalable, reusable and cost efficient. On the other end we leverage real-life test environments with actual LE's interacting with the system which has as an advantage that more realistic behavior can be included during testing. The hybrid simulation based testing technique we propose, combines the advantages of these two by enabling them to interact with the IoT system and with each other.

Finally, we notice that state-of-the-art simulation techniques are limited in the scaling capabilities required to orchestrate the real-time interaction of many thousands of LE's. Which is a requirement in order to rely on the hybrid testing

methodology that we propose. Therefore, we discuss various techniques that could improve the scalability constraints of modern simulation techniques.

The contribution of this paper is an extensive overview of the challenges that arise when testing large IoT applications at the system level. We present a novel hybrid simulation based testing approach, and introduce various solutions tot the challenges that arise when implementing this hybrid methodology. These challenges are mainly related to the IoT development pipeline, synchronization between real-life and simulation environment and the scalability constraints of modern simulation techniques. Furthermore, we map these solutions and challenges to a participatory sensing use case.
In the next section, we present the participatory sensing use case that serves as a running example throughout the paper. In section 3, an overview of classic IoT testing techniques is provided and the importance of local and emergent behavior is clarified. In section 4, we present various techniques that can be used to simulate behavior of local entities. Section 5 introduces the hybrid simulation based testing methodology and proposes various solutions to implementing this methodology. Finally, section 6 takes a deeper look at scalability constraints that arise when implementing large real-time IoT simulations.

2 Participatory sensing use case

To better explain the concepts presented in this paper we will use a running example throughout the remainder of this paper. The running example is based on the SeRGIo project, which is short for mobile sensing services for developing geospatial IoT application. It is a participatory sensing project that is currently being developed with multiple academic and industrial partners as a proof of concept. The goal of the project is to collect qualitative data from citizens or workers, such as mail(wo)men, by targeted distribution of small sensing tasks. SeRGIo is different compared to other participatory sensing frameworks in that they focus on the collection of qualitative data instead of quantitative data. An example of such data is the user's perception of safety in a particular neighborhood. A sensing task is typically received on the smartphone of the user and contains a small questionnaire or single question such as "How clean is the street?" or "What is the quality of the street bench in front of you?". SeRGIo will also take the spatio-temporal aspects of a user into account when sending tasks. This means that the system only sends tasks related to a certain place or area when a user is actually nearby.

It should be clear that testing this type of system poses significant challenges. This is mainly due to the fact that the functional aspects of the IoT system can only be tested with actual participants. In current state of the art, there isn't a clear approach to handle such test cases.

3 IoT testing

IoT applications are inherently heterogeneous systems. They consist of many different sensors, actuators, communication protocols and operating systems. This makes the testing of these systems a challenging task. In this section, we provide an overview of classical testing techniques used in IoT projects and identify the gaps that remain in current testing practices.

3.1 Classic testing of IoT

Given, that there is a strong cohesion between hardware and software in IoT projects, the testing approach used in state of the art Internet of Things projects are often based on best practices used in classic software or hardware development.

Software testing Software testing is mainly used for the validation of both functional and non-functional aspects of IoT middleware software or more low-level code such as the software logic deployed on IoT sensors and actuators [3]. Two major types of testing are described:

1) White box testing: the tests have full transparency to the inner structure of the software. Unit tests are often considered a white box testing approach. Within the context of IoT testing, this method is used to test the functionality of low-level pieces of code, such as single methods and classes.

2) Black box testing: With this testing method the software system is considered a full black box [2]. These tests have no knowledge of the internal structure of the system. Black box tests are concerned with more high-level aspects of the software, typically on the system-level. E.g. in SeRGIo we could apply a black box system level test to verify whether a user correctly receives a sensing task on his smartphone when he is located in a certain area that is below the data relevance threshold. In practice, these type of tests are difficult to run, especially when developing IoT applications. Because, as mentioned before, IoT systems typically contain many different sensors, actuators and software parts interacting with each other.

Within the context of this paper we only focus on the black box testing method.

Prototype setups Prototype testing is used to test the functionality of hardware components in a lab-based environment. These tests are often limited to single devices. Given the diversity of IoT systems and environments it is best to rely on more large scale test setups when testing entire systems. Various initiatives are described in literature to facilitate such large scale test beds. One example is the imec City of Things testbed in Antwerp [17]. Another example of such a testbed is the Smart Santander project
[25]. It is a city-wide, real IoT testbed. It offers many thousands of IoT devices readily deployed in an actual environment. Most of the testbeds described in

literature are ideal to test non-functional requirements of an IoT system, such as testing the interoperability between various devices and their communication protocols or operating systems.

Simulation based testing Instead of writing custom test cases, simulation models are used to interact with the system. Although real testing is often desirable, simulation testing is a more flexible and cost-efficient approach. It allows for a more controlled and reusable environment that can be tweaked much easier. Within the IoT domain simulation testing is most often used to test technical aspects of the system such as network related features, power consumption etc. The most well-known examples of such simulators are NS-3 [6] and Omnet++ [28]. Various IoT operating systems such as Contiki and TinyOs also offer their dedicated simulator, Cooja [11] and Tossim
[18] respectively. Some simulators are focused on testing more high-level IoT setups such as the iFogSim
[15] which is used to test IoT edge or fog architectures. Finally, D'Angelo et. al. demonstrate the use of the Gaia/Artis specifically to run large-scale IoT simulations
[10]. Also here, the focus is mainly on testing technical aspects such as power consumption and network utilization.

The problem however with the testing approaches described in this section is that they typically do not take the impact of behavior into account when testing IoT systems and are often focused on testing only non-functional requirements. Many of the classic testing practices isolate certain parts of the system and test it in isolation. We want to test the behavior of the system at the system level. Therefore, we look at the IoT system as a black box.

3.2 Role of behavior in IoT systems

To better understand the importance and the role of behavior in IoT systems it is important to clearly define different types of behavior. We differentiate between local behavior at the local or entity level and emergent behavior which arises at the global level of the IoT system.

Local entity behavior This type of behavior is exhibited by the local entity (LE) level of an IoT system. With the term local entities we refer to local actors that operate autonomously and have some type of behavior. This behavior can be very different ranging from a simple actuator that responds to certain inputs, a sensor forwarding inputs towards devices whose behavior is powered by an AI. But also human actors can be seen as local entities. Actually, any connected entity that is able to communicate with other entities in the IoT system can be seen as a local entity. In the Sergio use case, mail(wo)men and citizens are also LE's. We refer to these entities as LE's in the remainder of this paper. The local behavior exhibited by these LE's has an impact on the

overall behavior of the system. Therefore this behavior is an important part of the system and needs to be taken into consideration when evaluating the system.

Emergent behavior A special type of behavior in IoT that is gaining attention in the IoT community is emergent behavior. Emergent behavior is the overall, global behavior posed by a system. They inherently depend on the behavior and interactions of LE's with each other and with the environment. Mataric et al. define emergent behavior as a collection of actions and patterns that result from local interactions between elements and their environment, which have not been explicitly programmed [20]. It is loosely based on the emergent behavior that is observed in bird flocks where birds apply three local rules which result in emergent flocking behavior (e.g. remain x distance to neighbor birds). Roca et al. argues that this emergent behavior will lead to improved scalability, interoperability and cost efficiency of ultra-large-scale IoT systems as opposed to traditional approaches that strongly rely on extensive programming of explicit behaviors [23]. Emergent behavior typically originate from autonomous entities, with adaptive or evolving behavior, that are interacting with each other and with the environment. This type of behavior particularly benefits the IoT areas which require the interaction of an enormous amount of devices where relying solely on a centralized architecture is insufficient. Examples are such as smart power grids, autonomous car flocking and smart traffic lights.
In the context of this paper we are interested in testing functional aspects of IoT applications at the system level. Furthermore, we want to evaluate how we can include behavior when testing functional requirements of an IoT system. We believe that the dynamic impact of local entity behavior can not be neglected in IoT systems. As in many cases, there is a strong connection between human actors and the IoT system as a whole. The state-of-the-art (SoA) literature lacks a clear methods to validate, verify this behavior and its relationship to the IoT system as a whole. In the next sections, we present various techniques and list open challenges that arise when testing behavior in IoT systems. Furthermore, when testing emergent behavior in IoT systems, real-time interactions between LE's and the environment (e.g. a IoT middleware system) are required. Solutions to facilitate this are largely ignored in SoA literature. Later in this paper, we look at how we can leverage large-scale simulation techniques to do this.

4 Simulating behavior in IoT system testing

Many IoT services such as participatory sensing, adaptive traffic navigation and more, inherently rely on human behavior. Furthermore, some of them even depend on the behavior posed by people interacting with each other and with various IoT devices in order to provide qualitative services. As pointed out by D. Nunes in the context of cyber physical systems (CPS), most CPS

are however human-centric applications where humans are an essential part of the system and unfortunately, most of these systems still consider the human as an external and unpredictable element [22]. The same argument holds for Internet of Things systems and behavior posed by other local entities (LE) such as smart actuators and sensors.

For example, in the participatory sensing use case, human actors are responsible for collecting sensing tasks. These actors behave according to certain semi-predictable spatio-temporal patterns, e.g. a mail(wo)men walks the same route each day and an average citizen goes to work and returns back home at roughly the same time using the same route. Furthermore, the actors also add an element of stochasticity to the system. For example, not every Sergio user will always be able or willing to respond to sensing tasks.

One can not rely directly on human actors and readily deployed IoT LE's to evaluate the behavior of the system prior to when the system is fully operational. Therefore, simulation can be seen as an effective alternative to real LE's during the test phase. Opposed to solely simulating technical, non-functional aspects of IoT systems as described in the previous section, we propose the simulation of LE behavior in order to evaluate the functional requirements of IoT systems. These simulated entities (SE's) should be able to interact in real-time with the IoT system middleware, as if they were actual real-life entities. The following techniques can be used to model human behavior:

Explicit modeling The behavior of human actors or smart devices can be explicitly modeled. A technique often used for modeling the Internet of Things is agent based modeling (ABM). With ABM each IoT device or actor is considered an agent that is autonomous, dynamic and has the ability to interact with other agents or with the environment. These properties make ABM also ideal for modeling the behavior of humans actors [12]. Within the Sergio project we implemented an ABM approach to simulate citizens that navigate through the city. These citizen agents are able to walk a predefined route and broadcast their location to the middleware system. Furthermore, the agents were able to respond to task sensing requests based on a predefined probability estimate. This technique allows for an easy, cost-effective and reusable testing strategy compared to the classic testing approaches described in the previous sections. However, the level of detail in the modeled behavior is limited. It is therefore very likely that not all variations of behavior are included in the models which could lead to imperfections or errors in the demonstrated behavior.

Data replay Many IoT projects leverage existing datasets to replay behavior. For example, the participatory sensing system presented by M. Marjanovi et. al. uses a public dataset that contains discrete, timestamped GPS locations of 65 users [19]. For each user the data will be replayed at a predefined interval. The IoT middleware under test will perceive the incoming GPS logs as real-time behavior posed by an actual user. The same strategy could be used to test the Sergio project, to model the navigation behavior of citizens through the city. However, this would require the collection of qualitative data, which

is not always available depending on the type of data and the required quality. Furthermore, explicit modeling of behavior is still needed in many cases in order to respond to stochastic events. For example, with Sergio when a sensing task request is sent to a user, the system needs to have a response within a limited timeframe. Since the exact time of the creation of these events is not predictable upfront, we can not rely on replaying data event responses and thus, some logic is required to handle, process and respond to the event

Both explicit modeling and data replay can be effective strategies to model human behavior. They definitely can be used to include LE behavior in the IoT testing strategy. Compared to relying on a group of actual human testers, the simulation strategy is more scalable, more cost-effective and reusable. But, they require additional efforts and don't guarantee that the simulated behavior is realistic.

5 Hybrid testing

Using simulation can be an effective testing strategy. However, it is difficult to fully rely on it during the entire development process. As pointed out by D.A. Crisan et. al. the data generated by classic simulation methods, as presented in the previous section, cannot model precisely the traffic bursts and noise of real-world activities of people [8]. Real-world testing is still preferred as it will better represent the broad variations of human behavior. Therefore, we present a hybrid testing methodology. The methodology combines both the advantages of simulation, which is a cost-effective, scalable and reusable technique, with the advantages of real-life testing.
The concept of hybrid simulation has been described in related literature. Mainly, within the domain of structural construction simulation
[21]. It allows experiments to be conducted in which structural components with complex responses can be modeled experimentally and more well-known components can be represented within an analytical model. The motivation for using hybrid testing in the Internet of Things domain is similar. Combining complex behavior, represented by real-life testing with more homogeneous behavior represented by simulation. In more related work, Arora et. al. present a hybrid approach for their Kainsei, wireless sensing simulator [1]. However, in their work they consider the real world and the simulated world as two separate environments where no real-time interactions are required. Instead, data gathered in the real world need to be transferred manually to the simulated environment. In the following sections we present three novel solutions or methods that should be taken into account when implementing a hybrid testing approach.

5.1 Hybrid testing pipeline

The core of our methodology is based on a dynamic mix of both simulation and real-life testing as can be seen in Fig 1 below. In the early stages of development, functional testing of the IoT project will rely mainly on simulation and only for a small part on real-life testing. The amount of simulation will decrease when moving to later stages in the development pipeline. In the ex-

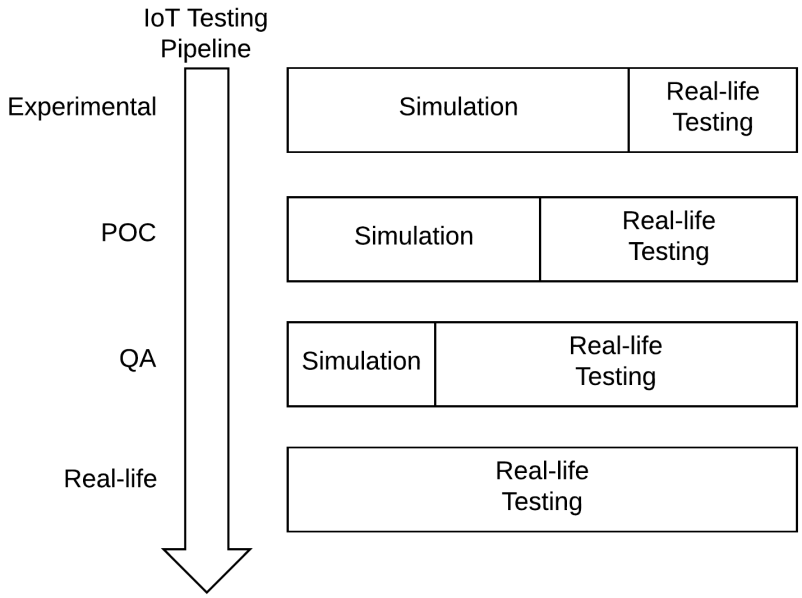

Fig. 1 IoT testing pipeline

perimental phase it makes sense to almost fully rely on simulation. At that point, fast development progress is key. Later phases move closer to deployment and correspondingly more accurate behavior is required. In the SeRGIo project, for example, a lot of focus during the early development stages was on building a stable simulator. Once the simulator was completed, we had a testbed environment that allowed us to easily experiment with various task distribution algorithms without worrying about the effort and cost that would occur when deploying a testbed in real-life. In later phases we focused on finalizing the most promising algorithms and gradually testing these in real-life environments.

5.2 Uniform communication interface

An important aspect of our hybrid testing methodology is that IoT middle ware system under test or other IoT entities are not aware whether the entity, which is generating specific behavior, is simulated or actually operating in real-life. The underlying method should be fully transparent as illustrated in Fig 2. The simulated environment should be able to interact with the middleware and with other entities in real-time. This has as a consequence that a clear

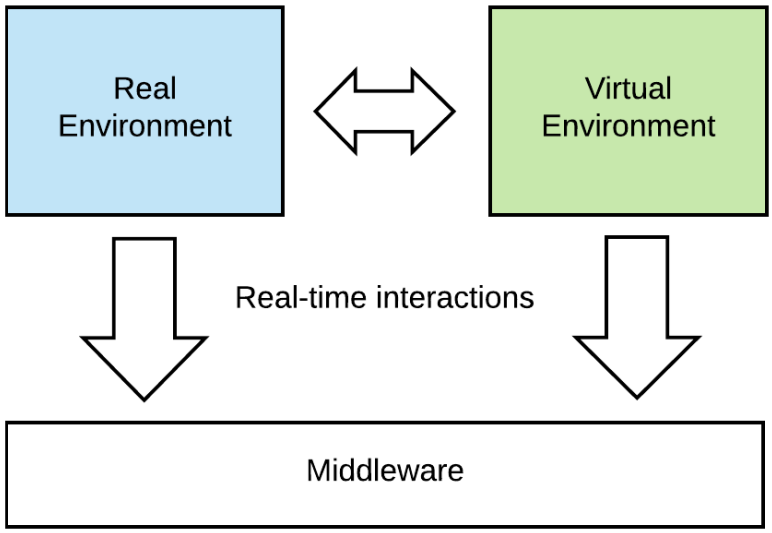

Fig. 2 Uniform interfaces

separation of concerns is required and that the communication interfaces are clearly defined and documented. The Sergio project relies on a variety of interface techniques such as REST and AMQP. In the context of IoT also COAP [26] and MQTT [16] can be used as more resource-efficient alternatives.

5.3 Synchronization challenges

By implementing a hybrid testing approach, there will be a mix of data generated by either simulated entities (SE) or real local entities (LE). It is important that there are no unnecessary inconsistencies between the data generated by the various sources. For example, in the Sergio use case it is possible that a SE responds to a sensing task request with a message that marks a specific area as

"dirty" while a real LE's would mark it as "very clean". As a result there will be a strong variation between the observed data points. This is specifically, a problem when testing emergent behavior. Let us for example assume that the Sergio project would adapt its behavior when a large variation in data points is detected in a certain area. Such variations could indicate that there is a lot of uncertainty about this area and additional sensing points are required to increase certainty. When more sensing points are added in the area, the load on the servers or edge devices located near the area will increase as a consequence. This in turn, could lead to a reconfiguration of reserved compute resources which is actually unnecessary and is only triggered as a consequence of the lack of synchronization between simulated and real LE's.
Clearly, a proper synchronization mechanism is required when implementing a hybrid testing approach. We present two techniques below. Synchronization can be implemented over multiple dimensions. We limit ourselves to synchronizing over the time and space dimension. In the context of IoT projects these are the most relevant dimensions. Within the space dimension we are concerned to maintain consistency between data sensed at a specific location. Within the time dimension we are concerned to maintain consistency between data sensed at a specific time. Preferably, both dimensions are take into account.

Proxy based synchronization: One of the requirements for implementing hybrid synchronization is that the IoT middleware is unaware whether the LE that is transmitting messages is real or simulated. Therefore, the synchronization mechanism cannot be implemented as part of the middleware. Instead, we propose a proxy synchronization mechanism as illustrated in Fig 3 below.

The proxy intercepts all messages send to the middleware. Messages originating from the real environment are directly forwarded to the middleware and also used to train a prediction algorithm which tries to match the space and time dimensions to the data values. Messages originating from the virtual environment will only contain a space and a time value. The proxy uses these values to estimate a data value that matches the data coming from the real environment. When no messages are received from the real environment, the prediction will be random. When time passes, and more and more messages are processed from the real environment, the predictions made by the proxy will better match the actual data distribution. An example can be seen in image 4 below where a Self Organizing Map (SOM) prediction technique is used to match the actual data distribution /citevan2012self. After training the SOM neurons are able to better match the actual temperature distribution. The illustration only shows the time dimension, also the space dimension should be used in practice but is now left out for visualization purposes. Suppose that a virtual LE sends a message at 12 am. The SOM will try to match this time value to the closest neurons, in this case the SOM will output a value somewhere near ten degrees Celsius and send it to the middleware. SOMs are a possible techniques, that is quite easy to implement, it allows for online learning and can quickly provide good results. But of course other prediction

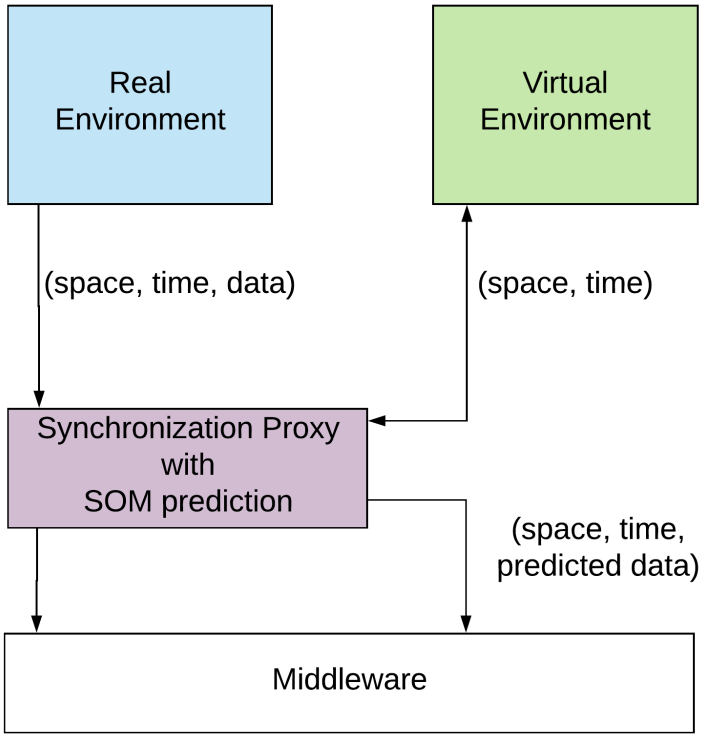

Fig. 3 Synchronization proxy

techniques can be used as well. In some specific cases it might even be useful to feed the sensor data originating from the virtual environment back to the real world that could impact the behavior and actuators of this real-life LE's.

Reducing synchronization requirement by space isolation: Another technique is to isolate certain geographical areas. These areas can then be limited to only serve real LE's or only simulated LE's. An example is illustrated in Fig 5 below. As a result, this reduces the need for an additional synchronization mechanism between real and simulated environment, like the one presented in the previous section. There will however, remain a mismatch in data variability at the border between areas (illustrated by a red line in the image below).

5.4 Evaluation of Hybrid Testing

To further motivate the advantages of implementing a hybrid testing approach we present our practical experiences gained from implementing a hybrid testing method in the SeRGIo project. During the early stages of middleware

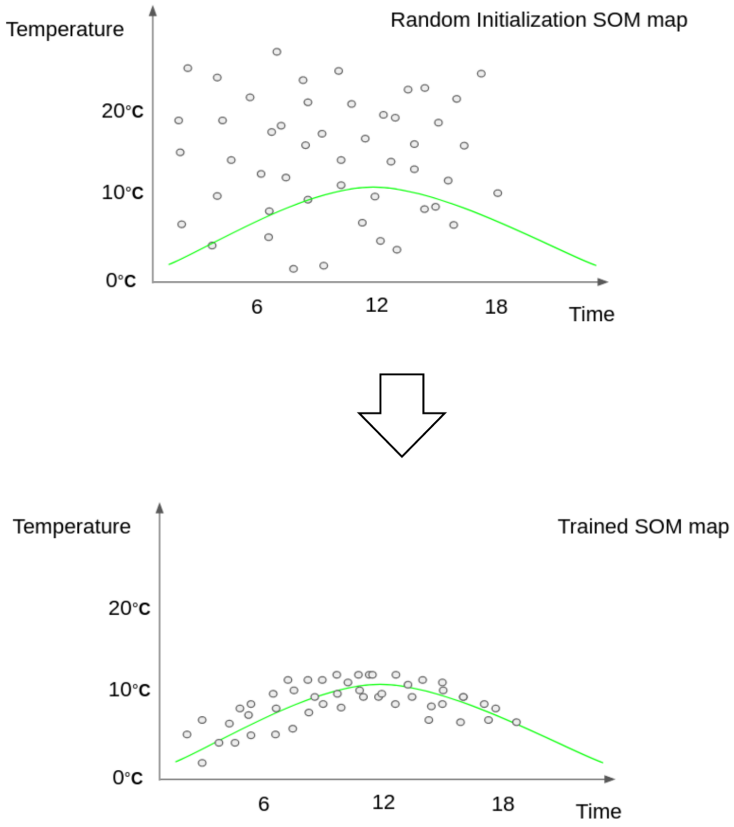

Fig. 4 Proxy prediction using a self organizing map

development we tested our sensing task distribution algorithm primarily in a simulation environment. One of the evaluation metrics was to verify the effort required by humans actors to perform a given sensing task. Based on that we could then calibrate the optimal refresh rate of sending out sensing tasks to obtain an optimal spread of data coverage in the city. It is hard, almost impossible to estimate this only on assumptions of human behavior modeled in a simulation environment. This is because we can't estimate upfront how much time a task would take and if people would either accept or reject the tasks. Instead we moved a stage further and tested the algorithm in a real environment using a basic smartphone application that was able to interact with the middleware under test and receive sensing tasks. An example of this task was "What is the cleanliness of the street at location X (which was in a max range of 400 meters)?". Testing this in real-life with actual human partic-

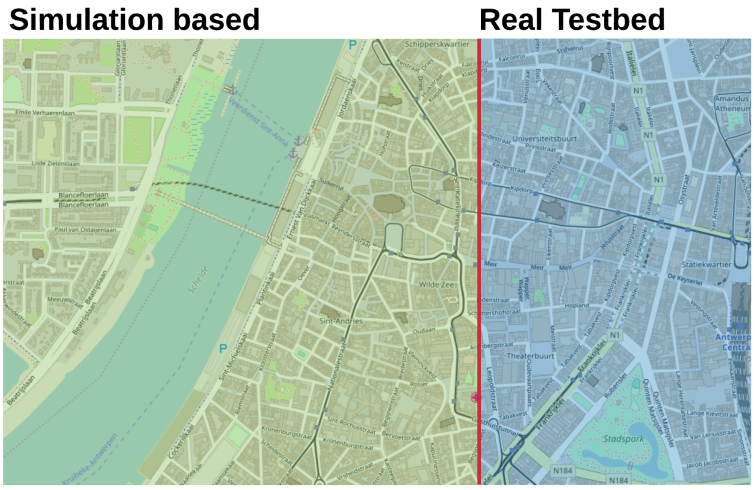

Fig. 5 Space isolation

ipators proved to be very valuable. It allowed us to better estimate how people would behave and as a consequence we were able to more accurately calibrate the sensing task distribution algorithm.

6 Towards large-scale real-time simulation

When testing large IoT systems the amount of LE's that need to be simulated will be very high. This is especially the case when simulating emergent behavior where a large amount of interacting entities are required. Most state of the art simulators are based on a monolith architecture which does not take advantage of parallelization or multi-server distribution that will be required to run many thousands of virtual LE's in parallel. An exception is the Gaia/Artis IoT simulator which implements the Parallel and Distributed Simulation (PADS) paradigm in order to run large scale Internet of Things environments. The parallel and distributed simulation (PADS) methodology [14] is IEEE standardized. It has proven to be an effective approach for running such a simulation over multiple nodes. With PADS a simulation is partitioned and executed among multiple distributed devices or Physical Execution Units (PEU's), this leads to a significant increase in scaling capabilities. Each PEU has a collection of logical processes (LP). An LP is a logical part of the simulation and contains various simulation entities (SE). A simulation entity represents an individual local entity. However, the PADS paradigm is not particularly optimized to support real-time interaction between simulation entities and real life prototypes, which is a necessity when implementing a hybrid simulation. In the remainder of this section, we present two optimization techniques that could

further improve the scalability capabilities of the PADS paradigm. Note that the findings of this work are based on recent work, described in more detail in a recently published paper that focuses on how to optimize the scalability constraints of state-of-the-art simulators [5].

6.1 Improving scalability through simulation model abstraction

In many cases the computational complexity of simulation models is much higher than needed. We propose to use model abstraction techniques [13] [7], which allows us to make a proper trade-off between accuracy and computational complexity. These abstraction techniques are then applied to one or more modeling formalisms. We differentiate between two model abstraction techniques that can be implemented.

Entity aggregation: The idea of entity aggregation is to combine local entities into a single high-level entity while maintaining the globally exhibited behavior. Rodriguez et. al. demonstrates that Neural Networks can be used to achieve this [24]. For example, instead of simulating multiple IoT sensors that measure temperature in the same geographical area, it might be better to replace all of them be a single SE as there won't be much difference between the observed temperature and it will definitely not impact the emergent behavior of the system.

State abstraction: This technique allows the abstraction of a single LE. The behavior of these individual LE's can be based on very complex models. We can use state abstraction to replace these very complex models with simpler approximations. An example technique described in literature is metamodeling. It allows to automatically learn the behavior of an LE and replace it with a surrogate model. Caughlin et. al. defines a metamodel as a projection of the original, high-fidelity model onto a subspace defined by new constraints or regions of interest [7]. Preferably, the created metamodel is more abstract and a less computationally complex version of the original model. Various machine learning techniques can be used to train such a metamodel.

6.2 Improving simulation resource partitioning

When applying the PADS methodology a major amount of performance loss is due to communication between simulated entities. This is especially the case when simulation entities (SE) need to interact with each other over the network when they're located on different PEU's. This computational cost of remote communication is much higher than the local communication cost between SE's located in the same PEU [4]. In this section we present two techniques that could reduce this remote communicational cost by trying to optimally distribute SE's:

Model partitioning using heuristics Most state of the art techniques that try to reduce remote communication cost of simulation entities rely on the implementation of heuristics to analyze the communication patterns of SE's and mark specific SE's as possible migration candidates. A consequence of this approach is that these algorithms lack a complete view on the global simulation state. Instead, they rely on limited data directly available on the PEU'S. D'Angelo proposes a sliding window approach in his work [9]. During a specific period, defined by the windows size, the ratio between remote interactions versus local interactions is evaluated for each SE. When the ratio exceeds a threshold value, the SE can be migrated to the LP it interacted with the most. This migration however occurs only when certain time has passed since the last migration of this specific SE.

Model partitioning using domain knowledge The heuristics approach is however limited by the fact that it does not take domain knowledge into account. This might lead to undesired side effects. For example, migrations might occur that need to be undone in a later phase. It could also result in a higher computational cost than the obtained gain which was only temporal. The heuristics presented above do try to prevent oscillating migrations by introducing a threshold that prevents immediate undoing a migration. However, another solution would be to inject domain knowledge that could prevent these unwanted migrations to occur in the first place. Furthermore, it could perform more optimal migrations that could not have been achieved by solely analyzing local communication patterns of individual SEs. Van Tendeloo and Vangheluwe demonstrate in their work that significant performance improvements can be obtained by injecting domain knowledge in their Python based distributed DEVS simulator PythonPDEVS [27]. They present the concept of activity prediction, which allows simulation entities to leak hints to the simulator kernel about their current and future activity and how they should be distributed. As a result, these hints can be used to decide when a SE migration should take place. This leads to an improved distribution of computational load across PEU's. A disadvantage is however that a level of transparency between the simulation modeling layer and the simulation core is lost. As a result, a trade-off needs to be made between a clean separation of concerns and overall simulation performance.

7 Conclusion

We presented in this work a novel hybrid simulation based testing technique. This technique allows IoT systems to be tested by orchestrating a real-time interaction between real-life and virtual local IoT entities (LE). The local behavior exhibited by these LE's eventually leads to a global emergent behavior. We identified a number of challenges that arise when implementing this hybrid simulation based technique. We presented an AI powered proxy based solution that could function as a synchronization mechanism. Furthermore,

we presented two possible techniques to solve the scalability constraints of state-of-the-art simulation techniques. One based on model abstraction and another based on the optimal resource distribution of simulation entities.

Acknowledgements SeRGIo is a project realized in collaboration with imec. Project partners are bpost, imec, Joyn and Nokia Bell Labs, with project support from VLAIO (Flanders Innovation & Entrepreneurship).

References

1. Arora A, Ertin E, Ramnath R, Nesterenko M, Leal W (2006) Kansei: A high-fidelity sensing testbed. IEEE Internet Computing 10(2):35–47
2. Beizer B (1995) Black-box testing: techniques for functional testing of software and systems. John Wiley & Sons, Inc.
3. Bertolino A (2007) Software testing research: Achievements, challenges, dreams. In: 2007 Future of Software Engineering, IEEE Computer Society, pp 85–103
4. Bononi L, Bracuto M, D'Angelo G, Donatiello L (2006) Proximity detection in distributed simulation of wireless mobile systems. In: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems, ACM, pp 44–51
5. Bosmans S, Mercelis S, Hellinckx P, Denil J (2018) Towards evaluating emergent behavior of the internet of things using large scale simulation techniques (wip). In: Proceedings of the Theory of Modeling and Simulation Symposium, Society for Computer Simulation International, p 4
6. Carneiro G (2010) Ns-3: Network simulator 3. In: UTM Lab Meeting April, vol 20
7. Caughlin D, Sisti AF (1997) Summary of model abstraction techniques. In: Enabling Technology for Simulation Science, International Society for Optics and Photonics, vol 3083, pp 2–14
8. Crisan DA, Radoi IE, Arvind D (2013) Coap-mediated hybrid simulation and visualisation environment for specknets. In: Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, ACM, pp 285–294
9. D'Angelo G (2017) The simulation model partitioning problem: An adaptive solution based on self-clustering. Simulation Modelling Practice and Theory 70:1–20
10. D'Angelo G, Ferretti S, Ghini V (2016) Simulation of the internet of things. In: High Performance Computing & Simulation (HPCS), 2016 International Conference on, IEEE, pp 1–8
11. Dunkels A, Gronvall B, Voigt T (2004) Contiki-a lightweight and flexible operating system for tiny networked sensors. In: Local Computer Networks, 2004. 29th Annual IEEE International Conference on, IEEE, pp 455–462

12. Fortino G, Gravina R, Russo W, Savaglio C (2017) Modeling and simulating internet-of-things systems: A hybrid agent-oriented approach. Computing in Science & Engineering 19(5):68–76
13. Frantz FK (1995) A taxonomy of model abstraction techniques. In: Proceedings of the 27th conference on Winter simulation, IEEE Computer Society, pp 1413–1420
14. Fujimoto RM (2000) Parallel and distributed simulation systems, vol 300. Wiley New York
15. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R (2017) ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Software: Practice and Experience 47(9):1275–1296
16. Hunkeler U, Truong HL, Stanford-Clark A (2008) Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In: Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on, IEEE, pp 791–798
17. Latre S, Leroux P, Coenen T, Braem B, Ballon P, Demeester P (2016) City of things: An integrated and multi-technology testbed for iot smart city experiments. In: Smart Cities Conference (ISC2), 2016 IEEE International, IEEE, pp 1–8
18. Levis P, Lee N, Welsh M, Culler D (2003) Tossim: Accurate and scalable simulation of entire tinyos applications. In: Proceedings of the 1st international conference on Embedded networked sensor systems, ACM, pp 126–137
19. Marjanović M, Antonić A, Žarko IP (2018) Edge computing architecture for mobile crowdsensing. IEEE Access
20. Mataric MJ (1993) Designing emergent behaviors: From local interactions to collective intelligence. In: Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior
21. Murray JA, Sasani M, Shao X (2015) Hybrid simulation for system-level structural response. Engineering Structures 103:228–238
22. Nunes DS, Zhang P, Silva JS (2015) A survey on human-in-the-loop applications towards an internet of all. IEEE Communications Surveys & Tutorials 17(2):944–965
23. Roca D, Nemirovsky D, Nemirovsky M, Milito R, Valero M (2016) Emergent behaviors in the internet of things: the ultimate ultra-large-scale system. IEEE micro 36(6):36–44
24. Rodriguez JD, Bauer Jr KW, Miller JO, Neher Jr RE (2008) Building prediction models of large hierarchical simulation models with artificial neural networks and other statistical techniques. In: Visual Information Processing, p 69780
25. Sanchez L, Muñoz L, Galache JA, Sotres P, Santana JR, Gutierrez V, et al (2014) Smartsantander: Iot experimentation over a smart city testbed. Computer Networks 61:217–238
26. Shelby Z, Hartke K, Bormann C (2014) The constrained application protocol (coap)

27. Van Tendeloo Y, Vangheluwe H (2014) Activity in pythonpdevs. In: ITM Web of Conferences, EDP Sciences, vol 3, p 01002
28. Varga A, Hornig R (2008) An overview of the omnet++ simulation environment. In: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p 60