*Mo*ρφευς: Simulate Reality for the Orchestration of Deployed Networked Embedded Systems

Richard Figura
CISS, Germany and University of Duisburg-Essen, Germany
r.figura@ciss.de

Matteo Ceriotti
University of Duisburg-Essen, Germany
matteo.ceriotti@uni-due.de

Sascha Jungen
University of Duisburg-Essen, Germany
sascha.jungen@uni-due.de

Sascha Hevelke
University of Duisburg-Essen, Germany

Tobias Hagemeier
University of Duisburg-Essen, Germany

Pedro José Marrón
University of Duisburg-Essen, Germany
pjmarron@uni-due.de

Abstract

Cyber-Physical Systems (CPSs) realise sensing and actuating infrastructures available for diverse applications with disparate requirements. These systems intertwine with the surrounding environment, making system performance difficult to foresee. During and after deployment, the ability to operate and validate the infrastructure is hampered by the limited visibility and the costs of testing alternative configurations. We propose *Mo*ρφευς, a framework able to analyse the performance of the target application in a faithful simulation. By calibrating the models with real measurements, *Mo*ρφευς can compute a virtual copy of the target system and reason about it. In particular, the visibility over the system state allows a thorough analysis of performance bottlenecks. Furthermore, the impact of alternative reconfiguration strategies can be tested beforehand. We instantiate our ideas in a prototype, with which we show that *Mo*ρφευς can suggest which nodes to activate in a physical infrastructure to control the application performance.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

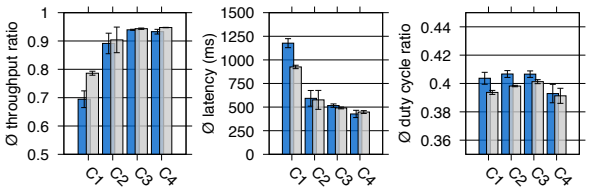
Design, Measurement, Performance, Reliability

Keywords

Cyber-Physical Systems, deployment, simulation

1 Introduction

Cyber-Physical Systems (CPSs) offer the unique opportunity to precisely observe the real world with a flexible technology. Due to their foreseen potentials, these systems are being deployed at larger and larger scales, e.g., to reach the extent of cities [11, 20]. Such infrastructures are meant to provide general-purpose monitoring services and serve a variety of applications, possibly changing at runtime. Similarly, dedicated infrastructures are being used in critical scenarios with strict requirements on, e.g., throughput, latency and lifetime. This realises a complex design space and, considering also the peculiar impact of each individual deployment scenario on the system behavior, it makes the analysis of alternative system configurations challenging.



(a) Throughput (b) Latency (c) Radio duty cycle

Figure 1. Application metrics for different real-life configurations (blue) and corresponding simulations (gray).

In Figure 1, we report the behavior of the CTP routing protocol [9] for different network configurations in our indoor CPS testbed (details about the setup are provided in Section 5.1). In particular, our goal was to identify the minimum set of devices necessary for the timely and reliable delivery of information about events happening in any room of our department. The different configurations correspond to the activation of different subsets of nodes in the network. Even small changes to the topology can have a significant impact on the network operation, transforming a malfunctioning system configuration ($C1$) into reliable ones ($C3$ and $C4$). In this paper, we focus on the problem of identifying which devices from a deployed infrastructure should be used by an application to fulfil its requirements in terms of throughput, latency and lifetime. In fact, the network graph alone generated by different selections of nodes is unable to describe the corresponding concrete application performance and, consequently, quantify the expected discrepancy between such network and the target requirements.

In the literature (Section 2), analytical approaches,

International Conference on Embedded Wireless Systems and Networks (EWSN) 2018
14–16 February, Madrid, Spain

ISBN: 978-0-9949886-2-1

e.g., [15], have tried to model various aspects of the system behavior in order to reason about its optimal configuration and operation. However, these approaches fail to quantify the specific performance of a system in its operational environment. As a consequence, system design and deployment resort to trial and error methods [32] in order to test and optimise the system behaviour directly in the target scenario. The lack of visibility into the system state and the possibly adverse conditions of the working environment make exploring the parameter space impractical. Furthermore, this approach requires operations where the system is deployed, increasing the maintenance costs and the time before the system can be reconfigured and adapted to changed or unsatisfied application requirements. Finally, software solutions [3, 33, 19] offer the possibility to define flexible protocols or online reconfiguration techniques. These methods are, however, specific to a given network stack and/or application and they typically assume that all the devices are partaking in the application, forcing any solution to adhere to the properties of the underlying physical infrastructure.

In this work, we introduce a simulation-aided planning framework, *Mo*ρφευς (Section 3). Our approach relies on the analysis of the application performance in a faithful simulation based on measurements coming from the target environment and system. By calibrating the models used in simulation with measurements taken in the target environment, *Mo*ρφευς becomes able to compute a virtual copy of the real system and reason about it. Specifically, by using simulation, it gains full visibility over the real system state. This allows to realise a thorough analysis (Section 4) of which devices provide a significant benefit or detriment to the application performance. Furthermore, reconfiguration strategies can be identified and alternative system configurations can be tested beforehand, quantifying with a reasonable accuracy the impact of each configuration on the real system.

In particular, our contributions are: 1) enabling faithful simulations through the calibration of communication models with real measurements; 2) identifying and assessing alternative application-agnostic reconfiguration strategies, based on the analysis of the simulation output; 3) supporting the analysis of the discrepancies between the selected reconfiguration and the corresponding behaviour of the real system in order to detect inaccuracies in the simulation models as well as network instabilities or system changes; 4) designing a concrete framework where the system information is analysed, possible reconfiguration strategies are evaluated, and reconfiguration actions are supervised during execution.

In the evaluation (Section 5) in our indoor testbed, we demonstrate that *Mo*ρφευς is capable of suggesting changes to the system infrastructure by accurately identifying areas with bottlenecks and proposing suitable reconfiguration actions. We confirm that controlled changes to the infrastructure can have a huge impact on the application performance and offer a larger degree of freedom in adapting the system to changing application requirements. Notably, *Mo*ρφευς is orthogonal to approaches that adapt software parameters. Once *Mo*ρφευς has chosen an appropriate infrastructure and topology, software approaches can still counteract dynamics of the environment and the system, before *Mo*ρφευς is able to apply alternative reconfiguration actions. From the discussion of *Mo*ρφευς (Section 6), we pave the way to its use in the holistic design of complex CPS networks (Section 7).

2 Related Work

The challenge of designing and planning low-power networked embedded systems has been addressed in the literature from different perspectives. We discuss them in this section, highlighting the unique contributions of our work.

Deployment Planning. The design of a system prior to deployment, e.g., the choice of the actual position of stationary sensors, is typically driven by application requirements and domain knowledge. During deployment, several tools can be exploited to understand the specific properties of the system in the target environment [8, 12]. Such information can be used to construct probabilistic models of communication and sensing quality for the different possible node placements. Based on this, it is possible to identify optimal sensor locations [15]. This data-driven approach shows good results because it can describe the system in its operational environment through real observations. However, the generality of the employed system models are inadequate to capture the complexity of the specific software services and protocols employed in practice. As such, these approaches are unable to match application requirements like throughput or latency resulting from a specific network software stack.

Even through factual knowledge acquired in the target scenario, designing a working system involves a lot of physical trial-and-error [32] during deployment. In particular, antenna orientations and small changes in the actual positioning might have huge effects on the final performance, justifying solutions where the complete parameter space is explored exhaustively [4]. Our approach, instead, exploits measurements collected in the target environment to model the behaviour of the physical communication layer. On top of this, we make use of the unmodified application software to profile the system and directly match the user requirements. In this manner, we can quantify the impact of alternative plans in terms of requirements satisfiability.

Configuration Planning. In addition to the parameters describing the physical deployment, an appropriate configuration for the services and protocols running in the system must be identified. A first indication of the expected behaviour can be gathered through simulation. While a variety of simulators can reproduce the behaviour of low-power networked systems, e.g., [21, 18], they reproduce generic setups, whose characterisation might significantly deviate from the actual final performance. For this reason, some work [29] have proposed the integration of measurements from the target scenario to exhaustively explore the network parameter space.

Considering the large variety of network protocols and their unique functioning, the selection of appropriate system services requires frameworks able to identify the best network stack fitting the deployment scenario [2]. Our work extends and complements these approaches by using simulation with real-world measurements from the deployment without requiring knowledge of the employed software solutions in order to perform system optimisation. In this way, we realise an application-agnostic framework nonetheless

able to match actual user requirements of a concrete system in its operational scenario. We achieve this not by changing the parameters of the running network stack but by selecting the set of nodes able to satisfy the application needs, exploiting an alternative possibility for system optimisation.

Performance Estimation. The approaches aforementioned rely, in different flavours, on models of the communication quality and features. The widely used log-distance path loss model [25] provides an analytical description of how the wireless signal degrades over distance. Its use is often justified by the low computational effort and the possibility to derive the environment-specific parameters from observations [23, 1]. Nonetheless, studies have demonstrated the deviations of the predicted performance from the one observed [14]. More detailed approaches, e.g., raytracing [27], typically require an accurate characterisation of the environment to precisely calculate the paths followed by radio signals between senders and receivers. While accurate, these models have high computational costs and require precise maps of the scenario and the obstacles in within.

Recognising the difficulty of estimating real system performance through simulation, the community has shifted to the use of publicly available testbeds [10, 6] as reference validation tools. Despite their ability to describe real system behaviours, they represent specific configurations and setups. As a result, the gathered insights are hardly applicable to other deployment scenarios and systems. In our work, we provide an analysis of how well the performance of a real system can be estimated through a simulation based on the emulation of the individual devices [21] and an extended communication model calibrated with real measurements. Through the analysis, we demonstrate that an appropriate simulation can offer a faithful description of the system behaviour. This allows to perform extensive monitoring and debugging as well as analysing reconfiguration strategies in a convenient virtual environment before applying concrete changes to the operational system.

System Monitoring and Debugging. The possibility of inspecting and debugging the network performance is crucial to ensure the correct functioning and identify possibilities for optimising the system behaviour. This is particularly relevant in the case of networked embedded systems with limited visibility in the system state and scarce resources. In the literature, it is possible to identify several approaches to instrument the running code in order to trace particular information, e.g., allowing the correct reply of the complete system behaviour a posteriori [16]. Different metrics can then be extracted to identify problems or simply compare the performance of similar solutions in different setups [24].

Monitoring and debugging systems online typically involves significant resources. However, it has been demonstrated that adaptation of common tools are possible [31]. Alternatively, useful metrics can be extracted without significant impact on the system performance [13] or through the deployment of secondary networks overhearing information and reconstructing views of the events in the network [26]. By reconstructing a virtual copy of the real system, our approach allows all these approaches to run transparently in an artificial environment with unconstrained resources. Furthermore, we show that in such setup it becomes possible to use relevant informations about the system performance that are available already without modifying or inspecting the actual source code running on each device.

System Adaptation. With detailed informations about the system, it becomes possible to adjust the behaviour of the network. In particular, models of the different network protocols can be then used to identify at run-time an optimal configuration and apply it to the system to counteract changes in the environment or in the application behaviour [33]. Other approaches can exploit knowledge and measurements of the behaviour of different network protocols in different conditions to adapt the configuration of the running system based on identified or learned rules [19]. Orthogonal to these approaches is the possibility to design or extend protocols to make them self-adapt to the current situation of the system and the environment with an appropriate tuning of the parameters [3]. Finally, if software solutions are not able to make the system match the expected system performance, it is possible to consider changes to the physical deployment by physically adding or relocating devices [30].

Our approach complements these solutions. In particular, we identify strategies to select which nodes could improve specific application metrics and validate if such selection matches the actual requirements through simulation. Once the selection is applied to the operational system, environmental dynamics could still affect the experienced behaviour. The aforementioned solutions could counteract such changes without triggering our optimisation scheme. These same schemes can, however, also exploit our framework by validating their adaptation schemes before applying it.

3 Design

In this work, we tackle the problem of designing an arbitrary complex network of low-power wireless nodes matching specific application requirements. In particular, we focus on scenarios where an infrastructure of a variety of sensing and actuating devices is available to be used by any application, potentially changing over time. The *Mo*ρφευς framework can nonetheless be extended to tackle also other types of system design and application deployment.

We now introduce *Mo*ρφευς as well as its underlying simulation engine. We also discuss the application and network metrics that we use as reference to study the system behaviour. This design is then validated to demonstrate the ability of the framework to accurately describe the behaviour of a real system, enabling the analysis and reconfiguration of operational systems (as presented in Section 4).

3.1 *Mo*ρφευς Framework

The literature on CPS deployments, as well as personal experience, agrees on the strong interdependence between the system behaviour and its operational environment. The design of a system must then take this factor into account in order to obtain the level of reliability required by the specific application at hand. In systems employed, e.g., in precise monitoring or in closed control loops, it becomes crucial to quantitatively measure the expected system performance and optimise the configuration based on a concrete analysis. This

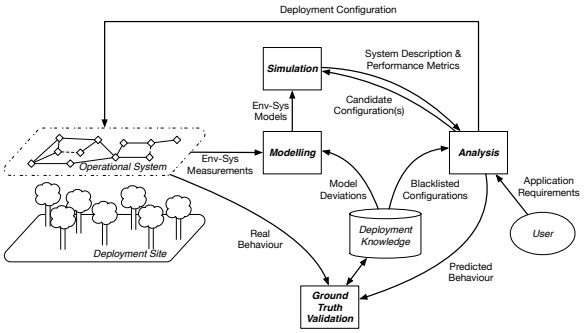


Figure 2. *Mo*ρφευς **framework and its use for deployment and reconfiguration planning.**

analysis is challenging to perform during deployment as exploring the complete design space in the operational environment is impractical, forcing trial-and-error [32].

As recently recognized [29], simulation can be used to support this analysis despite the typically argued discrepancy between synthetic and real-world behaviour. The key is the calibration of the models used in simulation based on actual information coming from the real system and environment. The result is a faithful virtual copy of the operational system, which offers full visibility over the system state and allows to analyse the configuration space.

Therefore, in *Mo*ρφευς, we start from real measurements coming from the deployment site. As depicted in Figure 2, such observations form the base on which to model both the scenario as well as the properties of the system once immersed in the target environment. An appropriate level of detail of such system and environment description enables the reproduction of the system behaviour in simulation, where we gain full control and visibility. Together with fitting network analysis techniques, the performance metrics monitored in simulation provide information used to analyse bottlenecks and identify alternative system configurations. Exploring candidate setups in simulation in parallel allows to find the most promising configurations for the operational system and analyse different trade-offs.

When applied, the overall application performance of this configuration can then be monitored. Deviations from the expected behaviour due to inaccuracies of the models used in simulation or physical changes of the operational environment and system can trigger *Mo*ρφευς to refine the deployment description and to compute an alternative configuration. Through this cycle, it is possible to validate system models and gather information of the impact of reconfiguration strategies on system behaviour, constructing a specific deployment knowledge. For example, it becomes possible to identify links or nodes that are detrimental to the application performance and blacklist them for a given period of time from the possible configurations or refine the simulation models with parameters fitting the observations.

3.2 Radio Model

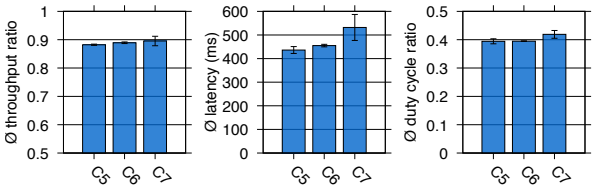
In order to build an accurate virtual copy of a running system, we base *Mo*ρφευς on the Cooja simulator [21]. The available support for hardware emulation [7], in fact, allows to replicate accurately the behaviour of individual nodes in isolation. In addition, typical CPSs realise the goals of the target application by exploiting distributed interactions among device. Therefore, it is crucial to define an appropriate model for communication. This is particularly the case for low-power short-range wireless, where different placements can significantly alter the properties of the resulting system. In addition, the peculiarity of the deployment scenario might have a significant impact, making the prediction of the individual link features challenging.

Among the different possibilities offered in Cooja, we decided to extend the basic DGRM model. This default communication medium behaves in such a way that two concurrent receptions at a node always result in a collision, interfering any ongoing communication. However, real-world experimentation shows a deviation from this simple behaviour requiring more complex medium descriptions in order to better replicate the functioning of a real network [17]. In *Mo*ρφευς, the actual link properties base on RSSI and PDR traces, gathered from the real network deployed in its target environment as in [29]. In addition, we describe interference and message collisions through a signal-to-noise-ratio (SNR) threshold in accord with previous studies [28], also taking into account possible capture effects [5].

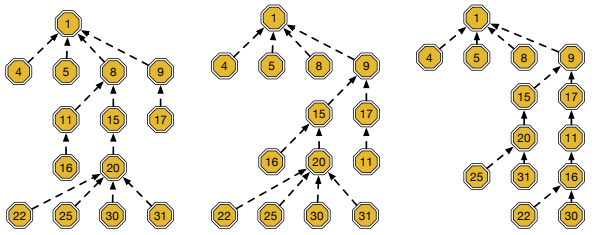
3.3 Metrics discussion

The analysis of the application behaviour and the identification of existing bottlenecks to use as input for possible reconfigurations can base on a variety of metrics. While the exhaustive exploration of the possible configuration space is out of the scope of this work, we target metrics and reconfigurations that can be computed without explicit information of the actual application implementation. Therefore, we limit ourselves to the traces that can be gathered from a deployed system about link properties, e.g., PDR and RSSI, as well as information about the application behaviour that the simulator, in our case Cooja, can extract about the simulated network without modifying the application binaries. While additional metrics and information can be included in the overall *Mo*ρφευς framework, we experiment specifically with analysis and reconfiguration strategies that are, therefore, independent from the deployed software stack.

With the collected PDR and RSSI informations regarding each link, we can execute in simulation the binary of the target application and monitor different accessible metrics. In particular, for traditional data-gathering applications along a routing tree rooted at a sink, it is possible to monitor how many messages are sent and received. By identifying the unicast transmissions, it is possible to distinguish between application packets and routing beacons. More complex pattern matching algorithms can also be implemented to further categorise different types of messages in more complex and heterogeneous scenarios. Latency can similarly be tracked under the assumption that the application data is not transformed on the path from the source to the sink. Finally, lifetime can be estimated through the radio duty cycle, i.e., the ratio of time the radio is kept active with respect to the overall simulation execution. These metrics can be computed individually for each node of the network or aggregated over the whole network where the application is run. For the applica-



(a) Throughput (b) Latency (c) Radio duty cycle



(d) Stable routing tree in *C*5 (e) Routing tree in the 1st run of *C*7 (f) Routing tree in the 2nd run of *C*7

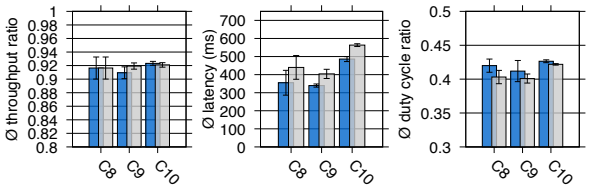
Figure 3. Application metrics and routing trees for different bootup orders and runs with the same set of nodes.

tion lifetime, a variety of estimations can be used, e.g., based on the highest radio duty cycle or the average one.

In addition to metrics of direct interest to the application that can directly be matched against user requirements, it is also possible to measure specific network properties. These metrics are relevant to study the functioning of the network, to understand the bottlenecks and identify corresponding reconfiguration actions. In particular, we compute the routing tree(s) of the paths followed by the data to reach the sink. By identifying the destination of unicast transmissions of each node, the data paths can be recognised and the overall routing structure employed by the application can be determined. Once this information is available, it is of interest to compute the tree edit distance [22], which counts the number of node removals and additions necessary to transform one routing graph into another. For convenience, in the rest of this paper we count a removal followed by an addition as one operation. Finally, by looking at successive traces of PDR and RSSI, it is possible to identify unstable links, whose quality significantly changes over time. This metric, which we refer to as link stability, is important to detect links that the system should avoid to increase stability and reliability. Moreover, the same metric can be exploited to identify links that the simulation is unable to reproduce accurately.

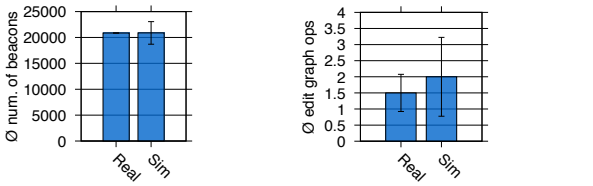
3.4 Fidelity Validation

Simulation inherently introduces randomisation between different system executions. This is justified by its typical use in the analysis of unspecific setups in order to gather generally applicable insights. Our goal is, instead, to study a specific system and identify bottlenecks and reconfiguration opportunities peculiar to such network. In particular, random seeds are used in simulation to slightly alter different executions and provide richer results. One consequence is that nodes are started at different times and in different orders. Assuming an identical system and software configuration, this can already have a significant impact on the observed system behaviour. For example, it is possible that



(a) Throughput (b) Latency (c) Radio duty cycle

Figure 4. Node-wise application metrics for simulations (grey) based on real-world measurements and same boot order of nodes in comparison to the reference real-world system performance (blue).



(a) Routing beacons (b) Graph distance from the reference base real-world routing tree of multiple real-world executions and simulations

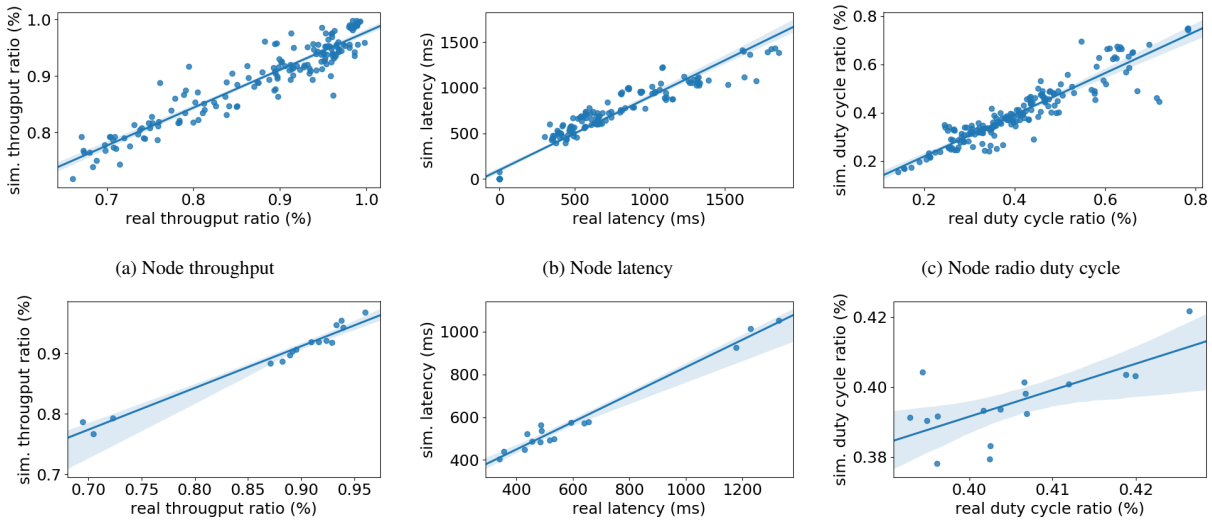
Figure 5. Routing metrics for simulations based on real-world measurements and same boot order of nodes in comparison to the reference real-world system performance (*C*8).

different routing paths are selected based on the timings of the beaconing and routing procedures.

We studied this effect in real setups by performing a series of repeated tests with nodes booted in different orders. We selected 14 devices in our testbed (described in Section 5.1) and randomly selected 3 different bootup orders and timings. Via serial, we controlled the boot of the different devices. The different configurations were executed one after the other in multiple runs to observe the behaviour in similar conditions. The results are shown in Figure 3.

The figures show that deviations are indeed possible and significant. Furthermore, different configurations might show more variation between different runs than others. By looking at the generated routing tree, the paths used to deliver data clearly differ from each other, explaining the differences in performance between different system configurations as well as the higher variability of results for configuration *C*7. Furthermore, the possible variations manifested by the same configuration indicate the need to treat explicitly the cases in which the network and the corresponding application performance are unstable. We address this point while discussing the reconfiguration strategies in Section 4.3.

The bootup order is one of the factors affecting the system behaviour, which can be practically monitored and controlled by looking at the local clock during typical time synchronisation procedures. However, also hardware-specific aspects can influence the network performance, e.g., clock drifts or the use of random number generators in the software itself. We exclude these elements from our study due to their intrinsic erratic behaviour. Further elements, e.g., ex-



(a) Node throughput (b) Node latency (c) Node radio duty cycle

(d) System throughput (e) System latency (f) System radio duty cycle

Figure 6. Linear regression and 95% confidence interval for the relationship between simulated metrics and corresponding real performance for all analysed network configurations.

Table 1. Pearson correlation factor over all experiments for the system-level and node-level application metrics.

Pearson correl.	*Throughput*	*Latency*	*Duty Cycle*
System-level	0.99	0.99	0.69
Node-level	0.94	0.96	0.92

ternal interference or environmental changes, require to extend both the communication model and the system analysis in the time dimension. This is an exciting development of our framework, which is part of our ongoing research agenda.

For testing the simulation fidelity, we compared the results measured in a real deployment with the ones obtained from corresponding simulations. We run a traditional data gathering application on 12 nodes in our testbed (the experimentation setup is explained in Section 5.1). Figure 4 shows the overall application performance for throughput, latency and lifetime metrics for three different configurations. Even though deviations are present, the general behaviour is preserved, supporting our goal of using simulation to perform the system analysis of the operational network in simulation.

Figure 5 demonstrates that also routing metrics, e.g., the number of beacons, are accurately reproduced in the artificial setup. We also compare the routing graphs resulting from multiple runs of the same network in simulation and in the real setup. Figure 5b shows that simulation manifests a routing tree at an average distance of 0.5 edit operations. This deviation is within the inherent variability of the network itself and therefore accurate enough to support our analysis.

In order to generalise our insights on the ability of the simulation to accurately replicate the real behaviour of the system, Figure 6 and Table 1 describe the correlation between simulated and real-life behaviour both for the overall system metrics as well as for each individual node in each tested configuration. Indeed, for all the metrics, all the points closely follow a linear relationship, with a Pearson correlation value close to 1. Just the system-level radio duty cycles differ significantly from the reality with a Pearson value below 0.7. Instead, the node-level simulated performance can still be used reliably in the analysis of the corresponding real system. These results ultimately support our choice of using simulation to study the behaviour of a real network.

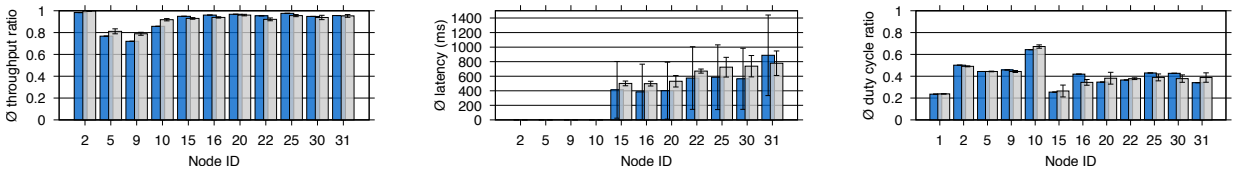
4 System Analysis and Reconfiguration

After discussing the approach, we turn our attention to the system analysis that *Mo*ρφευς enables and the possible corresponding reconfiguration strategies.

4.1 Reconfiguration with *Mo*ρφευς

The possibility to accurately reproduce the behaviour of a real system in simulation, as described in Section 3, offers the possibility to perform network analysis and optimisation without the constraints of the actual resources available in the deployed system. Once a virtual copy of the deployed network is available, it becomes possible to investigate at an arbitrary level of detail and complexity the functioning and performance of each individual device and of the system as a whole. In particular, the overall system target metrics can be exploited to identify configurations not satisfying the user requirements, detect unstable setups, or recognise discrepancies between real and simulated behaviours.

In any of the aforementioned cases, *Mo*ρφευς can further examine the system state in details in simulation, inspecting the behaviour of the individual nodes. Considering the gained visibility, it becomes possible to identify specific bottlenecks, which can both be used to determine the causes of instability or discrepancies from the expected performance and point to areas of the system with potentials for optimisations. In the former, blacklisting can hide the highlighted



(a) Throughput (b) Latency (c) Radio on-time

Figure 7. Node-level application metrics for simulations (grey) based on real-world measurements and same boot order of nodes in comparison to the reference real-world system performance (blue) for configuration *C*6.

nodes or links from the considered configurations. In the latter, instead, the analysis can guide the search for reconfiguration options, speeding up the procedure.

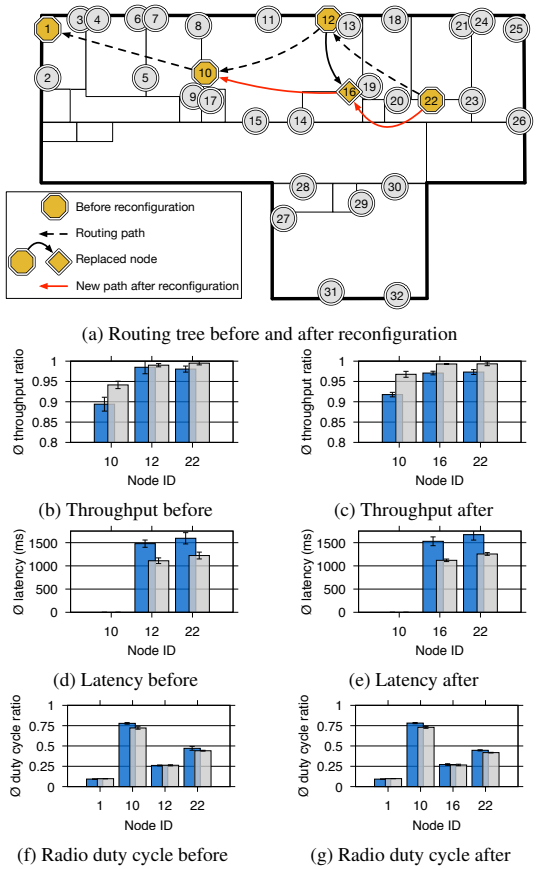
In *Mo*ρφευς, in fact, reconfiguration could in principle involve the exhaustive exploration of the feasible system configurations. As the analysed system grows, however, the amount of resources required to compute the necessary simulation runs might become prohibitive. Also considering the possibility to gain visibility into the network state, identifying the current bottlenecks can significantly reduce the alternative reconfiguration options to explore, thus increasing scalability. Once such setups are identified, they can be tested in simulations in parallel. If a candidate reconfiguration fulfils the application requirements, it can then be either applied or further analysed inside *Mo*ρφευς to recognise possible additional improvements.

4.2 Bottleneck discussion

Once the virtual copy of the real network has been executed in simulation, it is possible to analyse the different system and node metrics with the goal of identifying problems and bottlenecks. Figure 7 shows, exemplary for *C*6, the match of the node-level performance between simulation and real world. This confirms that the obtained information can be used to identify the bottlenecks with respect to specific target application requirements. An exhaustive description of possible bottlenecks is outside the scope of this paper, also considering their dependence on the specific application and scenario at hand. We now focus on different bottlenecks that we were able to study in our indoor experimental setup.

Weak Link Reliability. While the link performance of a given system is already evident through the performed monitoring, in simulation *Mo*ρφευς can identify the links utilised by the application. Depending on the routing protocol, the underlying topology might route data through links with low packet reception rates. This implies that retransmissions are required, increasing energy consumption and latency. With high error rates, also the network reliability can be affected, through either dropped messages or overflowing buffers.

High Node Workload. Even with reliable routing paths made by links with good reception rates, it is possible that the routing tree has crucial nodes in charge of supporting more workload than the average. In simulation, such a bottleneck can be identified either by recognising a high radio duty cycle of an individual node or by diagnosing a low throughput or a high latency of a set of nodes delivering data to the same ancestor node in the routing tree. Moreover, depending on both how well the nodes see each other and the density of the network, collisions can be present.



(a) Routing tree before and after reconfiguration

(b) Throughput before (c) Throughput after

(d) Latency before (e) Latency after

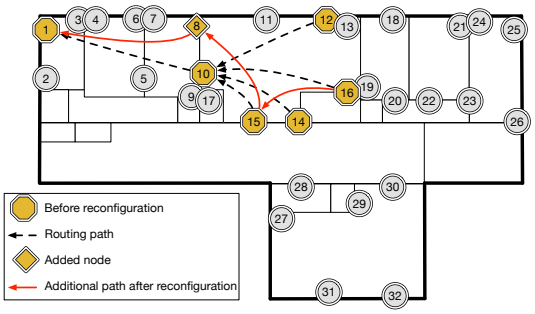
(f) Radio duty cycle before (g) Radio duty cycle after

Figure 8. Behaviour of a network with a weak link before and after the replacement of a node.

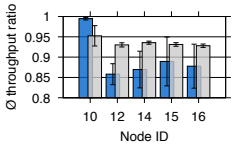
Long Routing Paths. The last case we focus on is of particular interest for large-scale networks. If leaf nodes are far from the sink, it is possible that a high number of nodes need to be traversed in order for the data to reach the final destination. As this happens, latency in particular increases. While routing protocols have become extremely good at identifying shorter and more effective paths exploiting existing nodes, our design framework adds the possibility to analyse alternative configurations with a different selection of nodes, unveiling an alternative parameter space for reconfiguration.

4.3 Reconfiguration Strategies

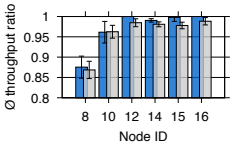
In our testbed setup, we tried to identify atomic examples of bottlenecks in order to analyse the possible applicable re-



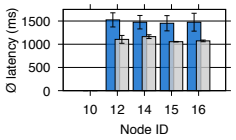
(a) Routing tree before and after reconfiguration



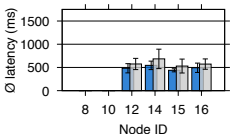
(b) Throughput before



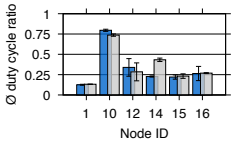
(c) Throughput after



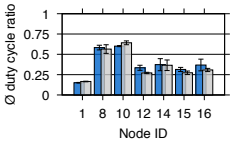
(d) Latency before



(e) Latency after

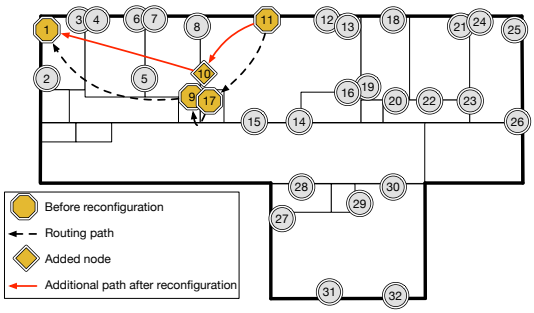


(f) Radio duty cycle before

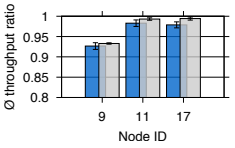


(g) Radio duty cycle after

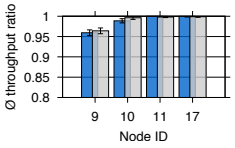
Figure 9. System behaviour of a network with high workload at a node before and after the addition of a node.



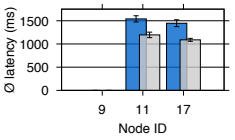
(a) Routing tree before and after reconfiguration



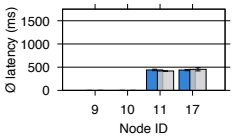
(b) Throughput before



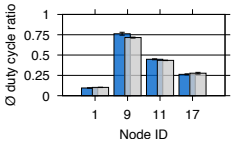
(c) Throughput after



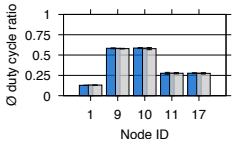
(d) Latency before



(e) Latency after



(f) Radio duty cycle before



(g) Radio duty cycle after

Figure 10. System behaviour of a network with high latency before and after the addition of a node.

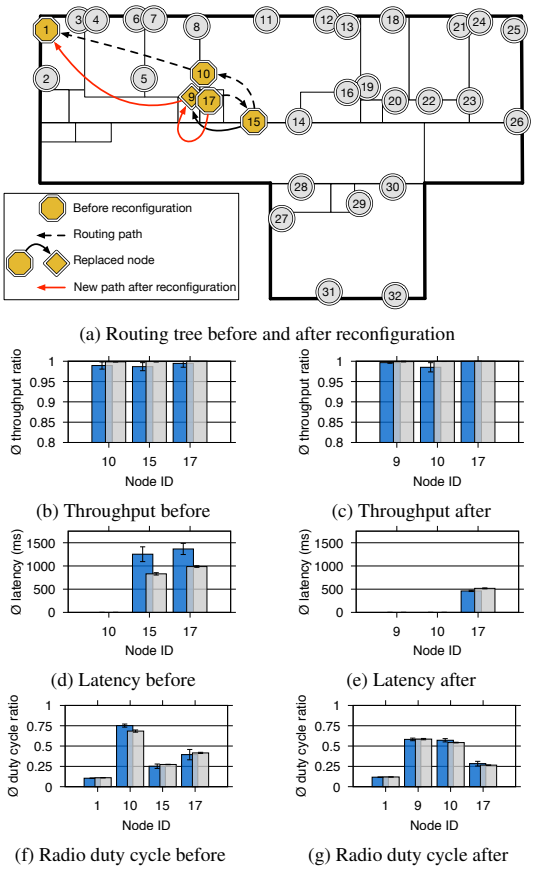
configurations and experiment with them.

Replace a Node to Repair a Weak Link. In Figure 8, two network configurations with the resulting performance are shown. In this case, the problem is a weak link between node 12 and node 10. Even though an alternative configuration can be identified in which node 12 is replaced by node 16, the simulation of such new setup shows no benefit in applying the proposed change. This demonstrates that reasoning in simulation about possible changes to a running system can avoid reconfigurations that are, indeed, unable to improve the performance. For this specific case, replacing a node is not sufficient, requiring new nodes to be added.

Add a Node to Distribute Workload. Another basic scenario that we could find in our testbed stresses node 10 as throughput bottleneck, as shown in Figure 9. In particular, it is possible to recreate a test case in which the resulting routing tree makes node 10 handle the traffic from 4 children. This has a clear impact on the network throughput. However, by looking in the surrounding of node 10, it is possible to identify the presence of node 8, which could take the workload of, at least, node 15. By running the simulation of the corresponding configuration, we can clearly identify that not only reliability improves, but also latency. On the other side, however, the number of nodes increases as well as the average radio duty cycle. With respect to this last aspect, it is worth noticing that lifetime nonetheless improves because the maximum radio duty cycle in the network is still lower than the initial configuration.

Add a Node to Reduce Latency. In Figure 10, we show a simple network where the generated routing tree has all the nodes inline. Latency is significantly high due to the length of the routing paths. However, adding a new node could address this situation by offering an alternative, shorter path to the sink. In fact, it is possible to identify a potential benefit in adding node 10, which has good links with nodes 11 and 1, the sink. The simulated performance, confirmed by real experimentation, demonstrates the significant benefit in latency, as well as in the other metrics.

Replace a Node to Be Close to the Sink. A similar scenario as the one previously tackled is depicted in Figure 11. In this case, the network throughput is high but with a high latency. Instead of considering adding a node, we try to replace one without changing the sensing coverage, i.e., considering only nodes in the same surroundings for possible removal or addition. Indeed, replacing node 15 (in the same area of 17) with node 9 (in the surrounding of node 10) significantly impacts



(a) Routing tree before and after reconfiguration

(b) Throughput before

(c) Throughput after

(d) Latency before

(e) Latency after

(f) Radio duty cycle before

(g) Radio duty cycle after

Figure 11. System behaviour of a network with high latency before and after the replacement of a node.

latency, being the added node closer to the sink than the removed one. Interestingly, this change has a clear impact only on one system metric, leaving the others unchanged.

5 Evaluation

After presenting atomic examples of the potential impact of our framework on guiding the overall system performance, we evaluate our approach in a testbed representative of an indoor setup. We present the scenario and describe the performed experimentation and the corresponding results.

5.1 Experimental Setup

Our experimentation was performed in an indoor testbed at our department. The system is made of 32 TelosB devices connected via USB cables to Raspberry Pies, which in turn are controlled from a central station, responsible also of gathering the information sent by each device via serial. The scenario is made of different rooms used as offices or laboratories, including a long corridor without devices.

Due to our previous experience, we based our experimentation on a traditional data gathering application based on the CTP routing protocol [9]. The system is configured to use node 31 as sink, channel 26 and power level 3. Each node in the active subset monitors environmental features, e.g., temperature and humidity, and sends a message every second,

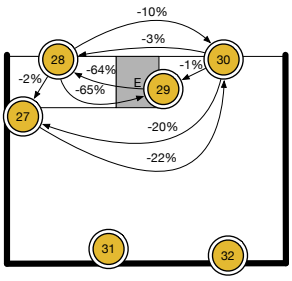


Figure 12. Links affected by the presence of the elevator at the same floor. The link demonstrating a change in PDR are depicted with the corresponding quality change.

with a wake-up interval of one second. Periodically, statistics about the device and protocols behaviour are reported over serial. The other nodes stay silent. The activation is driven by a central application, which also allows to specify the bootup order (in simulation and reality). The power level, message rate and wake-up frequency are selected to stress problems in the network.

Each system run takes 30 minutes, giving enough time for the routing protocol to stabilise. The relevant metrics are monitored after this initialisation phase is completed. Before each run, we execute for 30 seconds another dedicated application in charge of measuring the PDR and RSSI of the existing links. This information is then employed to configure the corresponding simulation environment, where the analysis of the system is performed in order to identify bottlenecks and evaluate reconfiguration options. In a real system, we expect this information to be gathered on demand or whenever a reconfiguration of the system takes place.

A full experimentation cycle, therefore, incurs the execution of the link monitoring application for 30 seconds on all the devices in the network. Each node is then reprogrammed with the data gathering application binary and only a target subset activated. The behaviour of the real network is then monitored for 30 minutes, during which we execute 5 simulation runs in parallel of the same application configuration and node selection. Based on the simulation, we identify bottlenecks and corresponding changes to the set of active nodes. We then re-run the same experimentation cycle changing the set of active nodes correspondingly. In this way, we use the real system execution to validate the fidelity of the simulation, while the simulation results are used as meant in the *Mo*ρφευς framework to reconfigure the system.

5.2 Blacklisting

As we want to target a stable application behaviour matching defined user requirements, blacklisting unstable configuration options is crucial to avoid foreseeable performance degradations once a configuration is applied. Moreover, filtering out such setups allows to speedup the analysis process. To this end, before considering system reconfigurations, we let *Mo*ρφευς process subsequent networking traces to identify possible sources of system instability.

In our specific scenario, the presence of an elevator has a significant impact on the links involving the nodes in its

surrounding. Upon detecting a variation in the application performance, the corresponding link informations can be used to identify significant changes in the network topology. As reported in Figure 12, the presence of the elevator can cause a decrease up to 60% PDR depending on the relative position of the devices. This analysis allows *Mo*ρφευς to identify blacklisting rules. In particular, configurations including any of the pairs of nodes $(28,29)$ or $(27,30)$ should be blacklisted. While adaptive protocols could address the same problem, they would first have to experiment the condition and trigger an adaptation every time the elevator reaches or leaves the floor, with corresponding transition phases in which the application requirements might not be satisfied. *Mo*ρφευς, instead, is able to avoid the situation once such condition has been experienced. Similarly, the same approach can be exploited to detect conditions depending on the time of the day, e.g., the interference caused by appliances such as microwave ovens or the presence of people, and schedule corresponding reconfigurations.

5.3 Impact of Alternative Configuration Strategies

We now turn our attention to the combined use of monitoring and simulation data in the *Mo*ρφευς framework to analyse system performance and evaluate reconfiguration options. As a reference use case, we take the collection of periodic data from the different offices at our department. To achieve that, we require that at least one sensor device in each room is active and that the data is collected at a central station corresponding to node 31. This recreates the most challenging scenario in our testbed, with a potentially high tree depth and different bottlenecks in order to reach the set of nodes at the opposite side of a long corridor without devices deployed. Furthermore, the aforementioned impact of the elevator is more significant. Being closer to the sink, it can affect the part of the network with the highest workload. In particular, we look into three case studies, manifesting different bottlenecks and possibilities for improvement.

5.3.1 Scenario 1: Add Node

The first network $C11$ satisfies the requirement of a system with one node in each office, employing the minimum number of devices to create a connected network. The map of the deployment and the corresponding application performance are reported in Figure 13. Throughput, latency and radio duty cycle are affected by the fact that the entire traffic has to be funneled through the connection between node 15 and node 28. This is then recognised as a bottleneck. Adding a node in this case should offer to the routing protocol multiple alternatives to convey the data to the sink, sharing the workload and offering multiple alternatives to identify the best routing paths in the most critical area.

One identified reconfiguration ($C12$) considers node 30, which would be able to take the workload of the nodes from the right side of the testbed and deliver data directly to the sink. The second possibility ($C13$) involves node 16, which could take the same role of node 30 but delivering data to node 28 better sharing the channel with node 15. The resulting simulated application performance underlies the significant higher benefit of $C12$ to improve throughput and latency.

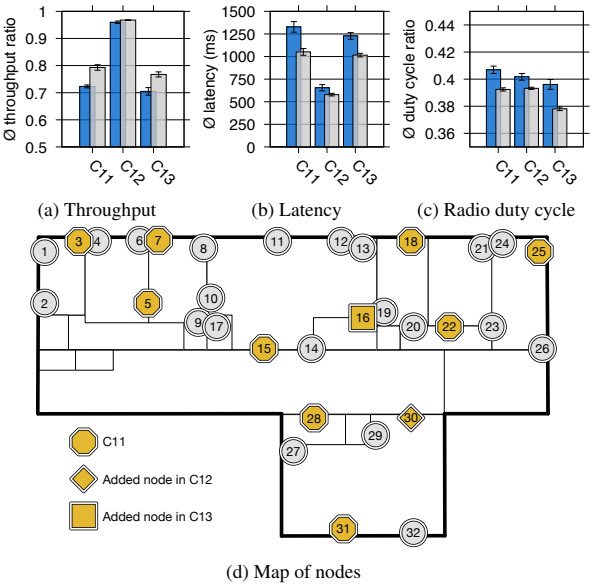


Figure 13. Application metrics behaviour for scenario 1.

For the radio duty cycle is, instead, more beneficial to select a configuration where the nodes handling most traffic are able to hear each other. These conditions allow the MAC layer to optimise channel sharing and minimise interference.

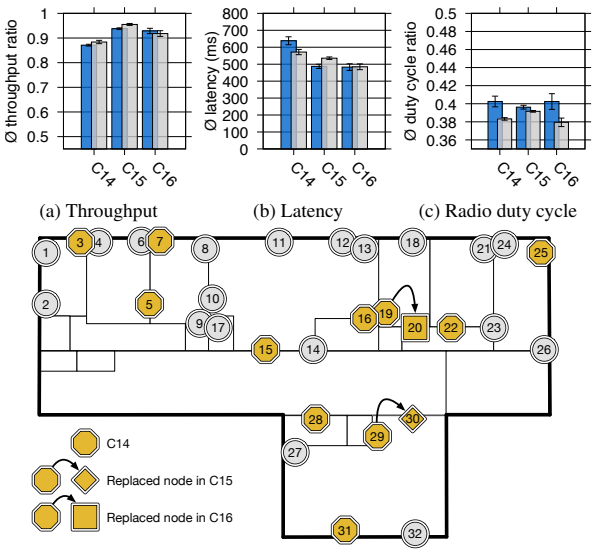
5.3.2 Scenario 2: Replace Node

Considering the impact of the elevator, we tested *Mo*ρφευς in a setup, shown in Figure 14, involving the link between node 28 and node 30, the most affected one by the presence of the elevator, without blacklisting it. In $C14$, node 29 can significantly affect the communication of node 28, which is responsible of handling the traffic of the whole part of the network above the corridor. The network is then affected by two problems, one is the link crossing the elevator that should be blacklisted and the high workload carried by node 28 along the connections with node 15 and node 16 (differently from the scenario discussed in Section 5.3.1, where only one link could be used).

In this case, *Mo*ρφευς identifies two options for changing active nodes. $C15$ considers avoiding the link crossing the elevator, which also creates a connection between the newly selected node 30 and node 19, thus splitting the traffic into two flows merging only at a sink. A similar characteristic is also achieved in $C16$ by selecting node 20. In this case, however, the unstable link crossing the elevator is preserved. The resulting application behaviour sees a clear benefit in terms of throughput and latency in both configurations. However, the unstable link still present in $C16$ causes a higher standard deviation and a higher radio duty cycle, supporting the choice made during blacklisting.

5.3.3 Scenario 3: Remove Node

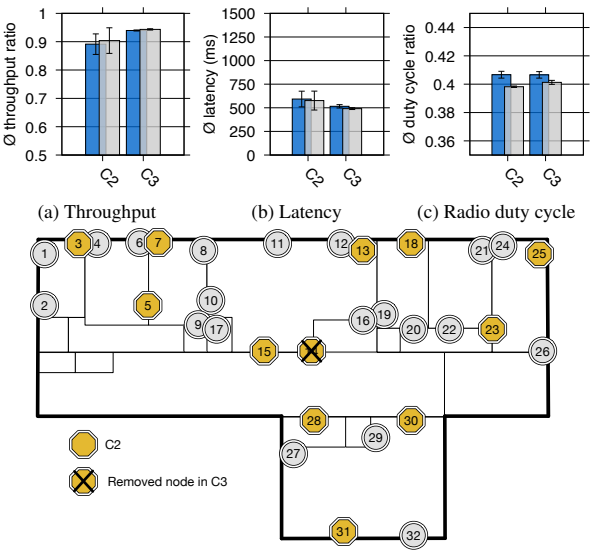
For the last scenario, shown in Figure 15, we tried to identify a redundant configuration offering multiple paths to the sink. In particular, we focused on the connection across the main corridor selecting node 14 and 15 on one side and node 28 and 30 on the other. All these nodes are able to



(a) Throughput (b) Latency (c) Radio duty cycle

(d) Map of nodes

Figure 14. Application metrics behaviour for scenario 2.



(a) Throughput (b) Latency (c) Radio duty cycle

(d) Map of nodes

Figure 15. Application metrics behaviour for scenario 3.

communicate with each other with the exception of the absent link between node 15 and 30. This setup would seem beneficial thanks to the offered redundancy. However, at the same time, it can affect the resulting application behaviour.

As shown initially in Figure 1, $C2$ experiences a throughput lower than 90%. Among the possible reconfigurations, the removal of a node would also decrease the overall system costs. In this case, node 14 becomes a candidate for removal considering that it is able to connect only to a subset of nodes that are neighbours of node 15. By testing this configuration in simulation, it is already possible to see that the throughput and latency improve, providing indeed a better and more stable performance. Interesting enough, *Mo*ρφευς could react at the cases when redundancy would be needed by reconfiguring the network looking at the resources available in the specific moment.

6 Discussion

In this section, we discuss the applicability of *Mo*ρφευς during system lifetime, its limitations and possible improvements as well as further developments.

*Mo*ρφευς **in the System Lifecycle.** Our approach bases on the availability of monitoring traces in order to create an accurate virtual copy of the real system. Therefore, nodes should be already deployed to allow *Mo*ρφευς to study the system behaviour. Alternatively, a pilot deployment should be carried out. Already available traces from similar scenarios are hardly applicable to our approach because each environment likely has different properties. To best meet the requirements of these scenarios, modelling techniques need to be introduced. In our currently ongoing work, we indeed demonstrate that it is possible to fully characterise an environment through in-site measurements and provided descriptions, e.g., maps. This allows not only to better match the aforementioned scenarios, but also to extend the reconfiguration space to also consider, e.g., possible benefits offered by freely relocating devices.

Our current framework, instead, applies preferably to generic network infrastructures where devices are already installed in the environment and available to different users and applications. In this scenario, we envision the iterative execution of reconfiguration steps where the addition, replacement, removing of active nodes is repeated until the user requirements are satisfied. The identification of unstable links and their temporary blacklisting allow to counteract environmental dynamics. In this process, *Mo*ρφευς might sacrifice optimality in order to reduce the search time and increase robustness against foreseeable changes in the environment.

Our approach is complementary to existing ones that require knowledge about the software running in the system and target the optimisation of the software network parameters. In fact, once *Mo*ρφευς identifies a network topology where to run the application, traditional software adaptation protocols can be employed to quickly reconfigure the network stack in case of system or environment dynamics. If it is not possible to counteract the degradation of the application performance, *Mo*ρφευς can newly be triggered.

Open Challenges. Simulation is a resource-intense process, in particular for large-scale systems. While we were able to execute the simulation runs for our scenario in real-time, the time required to reproduce the behaviour of systems can significantly grow with the number of devices and links, even more in the case of systems spanning a full city. Nonetheless, we believe that *Mo*ρφευς can further promote advancements in the field of simulation to support its applicability, considering its potential benefits for system design.

Similarly, the well-known discrepancy between simulation and reality might make *Mo*ρφευς take decisions detrimental to the application performance. Even if this is possible, a worsening of the application performance would then

trigger a new reconfiguration analysis. In this step, *Morpheus* would be able to blacklist the previously taken choice. Furthermore, this information could be used to identify which specific conditions manifested a discrepancy, highlighting specifically events and scenarios that the simulation models are not able to match. This allows to improve and refine simulation engines as well as gather new knowledge about the interplay between the system and the environment.

7 Conclusion

In this paper, we introduced *Morpheus*, a framework that allows to analyse real systems through accurate simulations, increasing visibility in the system state. This information can be used to reconfigure a running deployment tuning different parameters based on specific metrics and user requirements. In our work, we focus explicitly on metrics that do not require application or software knowledge to gain general applicability. We demonstrate that through this approach the application performance of a traditional application can be controlled by appropriately selecting which nodes to activate in a deployed network infrastructure.

8 Acknowledgments

The authors would like to thank Alexandr Krylovskiy for his preliminary analysis on the approach feasibility.

9 References

[1] A. AlSayyari, I. Kostanic, and C. E. Otero. An empirical path loss model for wireless sensor network deployment in an artificial turf environment. In *Proc. of ICNSC*, 2014.

[2] J. Ansari, E. Meshkova, W. Masood, A. Muslim, J. Riihijärvi, and P. Mähönen. Confab: Component based optimization of wsn protocol stacks using deployment feedback. In *Proc. of MobiWac*, 2012.

[3] C. A. Boano, K. Römer, and N. Tsiftes. Mitigating the adverse effects of temperature on low-power wireless protocols. In *Proc. of MASS*, 2014.

[4] M. Bocca, A. Luong, N. Patwari, and T. Schmid. Dial it in: Rotating rf sensors to enhance radio tomography. In *Proc. of SECON*, 2014.

[5] B. Dezfouli, M. Radi, K. Whitehouse, S. A. Razak, and H.-P. Tan. Cama: Efficient modeling of the capture effect for low-power wireless networks. *ACM Trans. Sen. Netw. (TOSN)*, 11(1):20:1–20:43, 2014.

[6] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda. Indriya: A low-cost, 3d wireless sensor network testbed. In *Proc. of TridentCom*, 2011.

[7] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón. Cooja/mspsim: Interoperability testing for wireless sensor networks. In *Proc. of SIMUtools*, 2009.

[8] R. Figura, M. Ceriotti, C.-Y. Shih, M. Mulero-Pázmány, S. Fu, R. Daidone, S. Jungen, J. J. Negro, and P. J. Marrón. IRIS: Efficient Visualization, Data Analysis and Experiment Management for Wireless Sensor Networks. *EAI Endorsed Transactions on Ubiquitous Environments*, 14(3), 2014.

[9] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis. Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. *ACM Trans. Sen. Netw. (TOSN)*, 10(1):16:1–16:49, 2013.

[10] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Network. In *Proc. of RealMAN*, 2006.

[11] ICE Gateway. https://www.ice-gateway.com.

[12] T. Istomin, R. Marfievici, A. L. Murphy, and G. P. Picco. Trident: In-field connectivity assessment for wireless sensor networks. In *Proc. of ExtremeCom*, 2014.

[13] M. Keller, J. Beutel, and L. Thiele. The problem bit. In *Proc. of DCOSS*, 2013.

[14] D. Kirov, R. Passerone, and M. Donelli. Statistical characterization of the 2.4 ghz radio channel for wsn in indoor office environments. In *Proc. of ETFA*, 2016.

[15] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Robust sensor placements at informative and communication-efficient locations. *ACM Trans. Sen. Netw. (TOSN)*, 7(4):31:1–31:33, 2011.

[16] O. Landsiedel, E. M. Schiller, and S. Tomaselli. LibReplay: Deterministic Replay for Bug Hunting in Sensor Networks. In *Proc. of EWSN*, 2015.

[17] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *Proc. of IPSN*, 2007.

[18] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proc. of SenSys*, 2003.

[19] D. Minder, M. Handte, and P. J. Marrn. Tinyadapt: An adaptation framework for sensor networks. In *Proc. of INSS*, 2010.

[20] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, and J. B. andMatt Welsh. Citysense: An urban-scale wireless sensor network and testbed. In *Proceedings of the IEEE Conference on Technologies for Homeland Security*, 2008.

[21] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proc. of SenseApp*, 2006.

[22] M. Pawlik and N. Augsten. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56(Supplement C):157 – 173, 2016.

[23] G. E. Perez, A. Alsayyari, and I. Kostanic. Comparison of the propagation loss of a real-life wireless sensor network and its complimentary simulation model. In *Proc. of HPCC-CSS-ICESS*, 2015.

[24] D. Puccinelli, O. Gnawali, S. Yoon, S. Giordano, and L. Guibas. End: A topology-aware collection metric for sensor networks. In *Proc. of SenSys*, 2010.

[25] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2nd edition, 2001.

[26] M. Ringwald, K. Römer, and A. Vitaletti. Passive inspection of sensor networks. In *Proc. of DCOSS*, 2007.

[27] K. R. Schaubach, N. J. Davis, and T. S. Rappaport. A ray tracing method for predicting path loss and delay spread in microcellular environments. In *Proc. of VTS*, 1992.

[28] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proc. of SenSys*, 2006.

[29] M. Strübe, F. Lukas, B. Li, and R. Kapitza. DrySim: Simulation-aided Deployment-specific Tailoring of Mote-class WSN Software. In *Proc. of MSWiM*, 2014.

[30] G. Tuna, V. C. Gungor, and K. Gulez. An autonomous wireless sensor network deployment system using mobile robots for human existence detection in case of disasters. *Ad Hoc Netw.*, 13:54–68, 2014.

[31] J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: A comprehensive source-level debugger for wireless sensor networks. In *Proc. of SenSys*, 2007.

[32] X. Zheng, C. Julien, M. Kim, and S. Khurshid. Perceptions on the state of the art in verification and validation in cyber-physical systems. *IEEE Systems Journal*, PP(99):1–14, 2015.

[33] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele. ptunes: Runtime parameter adaptation for low-power mac protocols. In *Proc. of IPSN*, 2012.