Research online Publishing www.ronpub.com

Open Access

Open Journal of Internet of Things (OJIOT)
Volume 7, Issue 1, 2021
http://www.ronpub.com/ojiot
ISSN 2364-7108

Network Metrics Detection to Support Internet of Things Application Orchestration

Thamires C. Luz[A], Cíntia B. Margi[A], Fábio L. Verdi[B]

[A]Universidade de São Paulo, Av. Prof. Luciano Gualberto, tv 3, 158, São Paulo, Brazil, {thamires.luz, cintia}@usp.br
[B]Universidade Federal de São Carlos, Rodovia João Leme dos Santos, Sorocaba, Brazil, verdi@ufscar.br

Abstract

Software Defined Wireless Sensor Networks (SDWSN) play an important role to serve as an infrastructure to Internet of Things (IoT) applications. In order to improve coverage, reduce costs, and make better use of the available resources, sharing the infrastructure among multiple applications is necessary. Works in the literature aim to enable resource sharing by allocating applications dynamically according to the resources available on the node. However, these works do not monitor if a node stops complying with application requirements once the application is allocated. Thus, network metrics detection is essential to identify nodes that are not able to comply with the application requirements. In this paper, we present the IT-SDN Manager architecture which is composed of a monitoring module and a resource orchestrator. The monitoring module monitors the network metrics, enabling the orchestrator to identify nodes that reach a certain threshold for energy available and packet loss. This threshold configuration depends on the metric characteristics. For packet loss, we present a study showing how it should be defined according to the network size and applications executed in the network. In order to evaluate the orchestrator detection rate, we set two application requirements to identify nodes that reach 90% of available energy and packet loss greater than the obtained threshold for each scenario studied. Results from the simulations executed show that the resource orchestrator detects all the nodes that reach the available energy threshold, and at least 85%, with an average of 97%, of the nodes that reach the packet loss threshold.

Type of Paper and Keywords

Regular research paper: *management, network metrics detection, software defined wireless sensor networks*

1 Introduction

Internet of Things (IoT) applications come across in different fields, such as Smart Industry, Mobility, Healthcare and Smart Cities. Each application has different characteristics and requirements, what makes devices and communication patterns behave differently. For instance, an application designed to detect an emergency, requires nodes to detect the event and transmit the information with low delay. On the other hand, applications monitoring pedestrians in a certain region require more throughput [8].

Consider a Smart Campus scenario. The sensor devices should be placed in specific locations according to each application goal. For instance, one application could target human movement detection, while other application would focus on environmental monitoring,

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2021)* in conjunction with the VLDB 2021 conference in Copenhagen, Denmark. The proceedings of VLIoT@VLDB 2021 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

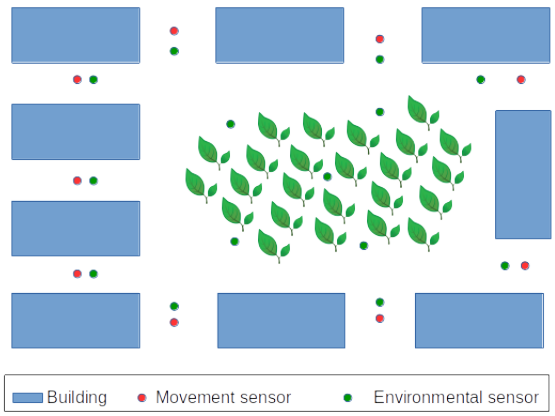


Figure 1: Smart campus illustration

and a third one on monitoring water usage and waste capacity [8]. Figure 1 depicts a small area with a couple of buildings and different sensor devices. We noticed that, in several locations, movement sensors and environmental sensors are close together. Often times, the same device is able to provide data for both the movement detection and the environmental monitoring applications. The costs to implement the sensing grid and monitoring a large area increases if we consider that each application would run on different hardware. Furthermore, there might be areas where only a transit network is needed to connect different campus area. Thus, sharing the sensing and communication infrastructure could considerably reduce the costs. However, to do so, it is necessary to consider the applications requirements and the hardware resources.

Software Defined Wireless Sensor Networks (SDWSN) are part of the infrastructure used to sense data for IoT applications [14]. With the increasing number of IoT applications, sharing the sensor nodes among multiple applications is necessary to reduce the investment in the infrastructure, and improve coverage and resource utilization [1]. However, sharing the devices could lead to inefficient resource utilization and network overload, given the Wireless Sensor Networks (WSN) resource constrained nature [11]. Moreover, without resource orchestration, the application requirements could not be provided by all nodes. Therefore, the use of efficient resource allocation and management techniques is of pivotal importance [13].

Virtualization is one way to accomplish IoT resource sharing. In this approach the virtual devices are implemented in the physical device according to its resource availability. SD-IoT controller [11] aims to enable resource sharing by orchestrating the sensor nodes with dynamic application allocation. While SD-IoT [11] relies on a controller behaving as an orchestrator and their own protocol, Li and Zong [6] use the controller to verify the energy available in the nodes to determine if multiple applications could run in the same sensor nodes.

However, despite the fact that a sensor node is capable of handling multiple application on its deployment, one can not guarantee that the node will still comply with the application requirements later on. For instance, available energy decreases, noise might compromise a certain region of the network, forwarding queues might become full, network dynamics might change. To the best of our knowledge, current works do not monitor and act when a device allocated with multiple applications stops complying with predetermined application requirements. Thus, detecting nodes that do not comply with application requirements becomes of critical importance to orchestrate resource utilization, even after application allocation.

In order to fill this gap, we propose a resource orchestrator that periodically monitors the network and detects when a node stops complying with application requirements. Unlike previous works that use the control plane to manage the network metrics, we propose and evaluate an architecture whereby the management plane is decoupled from the control plane. Therefore another entity handles the management information, the IT-SDN Manager. This manager has a REST API that enables the applications to configure their requirements.

The orchestrator detects non-compliant nodes based on a given threshold configured through the REST API. In this work, we study application requirements related to the energy level and packet loss in the nodes. However, it is difficult to determine what threshold to use for packet loss, since this depends on the network load and behavior. Hence, we evaluate how applications and network size impact the packet loss in order to identify the threshold that the orchestrator should use to monitor.

The contributions of our work are:

1. The design, implementation, and evaluation of an architecture whereby the management plane is decoupled from the control plane. Therefore, the manager entity in our architecture is responsible for all metric monitoring and management information unlike previous works that use the control plane to manage the network metrics;

2. The design and implementation of the resource orchestrator module to detect nodes on the network that do not comply with application requirements. The proposed resource orchestrator periodically monitors the network to analyze if the nodes comply with the application requirements; and

3. A study about how applications and network size impact the packet loss, in order to determine the most suitable packet loss threshold for the resource orchestrator detection.

To validate our proposal, we evaluate the impact of the manager overhead (in terms of delivery rate, delay and energy consumption) and the orchestrator module detection performance. We use the IT-SDN [2] [3] framework to enable SDWSN and the monitoring module [7] in the management plane to obtain node metrics for the resource orchestrator. The monitoring module monitors energy level and packet loss metrics to enable the orchestrator to identify nodes that reach 90% of available energy, and nodes that lose more than the given threshold for application and network size. Our results show that the resource orchestrator detects the nodes that reach the available energy threshold, and at least 97% of the nodes that reach the packet loss threshold.

The paper is organized as follows. Section 2 presents related work for management and orchestration. The IT-SDN Manager and resource orchestrator module are presented in Section 3. Section 4 describes the methodology and experiments designed. Section 5 presents the study for packet loss behavior, while the results are discussed in Section 6. Lastly, Section 7 presents conclusions and future works.

2 Related Work

SDWSN benefits from the Software-Defined Networking (SDN) paradigm to improve network management and network configuration. The first works [10] with the SDWSN paradigm focused on achieving network management to improve energy consumption. After that, research focus shifted to resource sharing and application requirements [1]. Thus one must understand how to share the same sensor nodes between multiple applications without compromising network performance.

SDNMM [9] is a modular management system for SDWSN. It is implemented using IT-SDN framework and a management service interface (MSI). MSI obtains the application configuration, be it either tasks or policies, and configures the nodes to execute it. The controller is responsible for routing and cluster management, as well. The SDNMM resource allocation module is capable of adapting the packet rate based on resource availability and improving task allocation based on node capability.

Besides SDWSN, some works use sensor virtualization to virtualize multiple sensors in the physical node and then enable resource sharing. The SD-IoT [11] controller is part of an architecture that enables resource allocation dynamically by virtualization. This controller acts both for routing rules and application requirements orchestration. S-MANAGE protocol transfers the application tasks to virtual nodes that translate the commands to physical devices. This architecture enables the same sensor nodes to be used by multiple applications considering the application requirements and the available node capacity. However, this architecture does not detect when a node stops complying with the application requirements, which could lead to network overload.

Li and Zong [6] propose a resource allocation strategy to meet the requirements of multiple concurrent applications in the same node. This architecture uses a central controller to schedule the tasks in the virtualized node. According to the application request, the central controller provides a service for the specific application based on the given resource allocation strategy. Therefore, the node can serve multiple applications at the same time considering the application requirements.

SensOrch [4] is a resource orchestration scheme that aims to enable resource sharing for multiple applications considering the Quality of Service requirements. To ensure efficient resource utilization, SensOrch assigns the physical sensor nodes to serve virtual sensor nodes taking into account the application requirements and the limited capacity to serve them.

These works present an architecture to enable resource sharing through task allocation considering the application requirements. However, after the task is allocated, neither of them detect when a node stops complying with application requirement. In addition to ensuring that the node has resources available at the application deployment, it is important to monitor if the node continues to comply with application requirements after the application allocation. Our work periodically monitors the sensor nodes to fill this gap. Therefore, our work differs from the previous in three main aspects. First, we periodically monitor node metrics to orchestrate the resources, and check if the nodes comply with the predetermined application requirements. Second, the controller is only responsible for routing and topology, and the manager entity is responsible for the management plane that could be active or not. Third, we do not allocate applications dynamically, but we detect when a node stops complying with the application requirements.

3 IT-SDN MANAGER

In general, the SDN approach separates the control and data planes, enabling the devices to be programmed with forwarding information. In the control plane, the SDN controller obtains information about the network topology, calculates routes, and sets all the flows used to forward data and control messages. When a node does not have information to forward a given packet, it requests to the SDN controller.

We used the IT-SDN framework [2] [3] to control the network topology, routing and enable SDWSN nodes. IT-SDN has a clear separation of protocols to achieve southbound communication (SB), neighbor discovery (ND) and controller discovery (CD). The SB protocol is used for the communication between the SDN controller and the network devices. The ND protocol is used to maintain node neighborhood information. And, CD protocol is responsible for finding the next hop to reach the controller. The IT-SDN monitoring module [7] is responsible for collecting node information, and was implemented as part of the controller.

RFC-7426 [12] recalls that "that the role of the management plane has been earlier largely ignored or specified as out-of-scope for the SDN ecosystem". The main characteristics used to differentiate the control and management planes concerns the mechanics, capabilities, and needs of each respective interface. They are as follows: timescale, persistence and locality. Also, as discussed in the RFC-7426 [12], "in the context of the CAP theorem, if one considers partition tolerance of paramount importance, traditional control-plane operations are usually local and fast (available), while management-plane operations are usually centralized (consistent) and may be slow." Thus, we propose the IT-SDN Manager architecture to separate the management functions from the SDN controller, and implement improvements in the management plane.

The IT-SDN Manager architecture is based on the RFC-7426 [12] which is illustrated in Figure 2.

The sensor nodes, depicted at the bottom, sense and transmit data to the sink according to the application characteristics. This communication is implemented in the IT-SDN framework using the SB, ND and CD protocols, as described before.

In the new architecture proposed in this paper, we design and implement a manager entity that is responsible for the management plane. The manager is responsible for configuring the nodes and monitoring the node metrics. Thus, the IT-SDN monitoring module that was originally implemented in the controller was moved to the manager entity. The node sends a management packet with metric values to the manager every minute and these values are stored in the manager database.

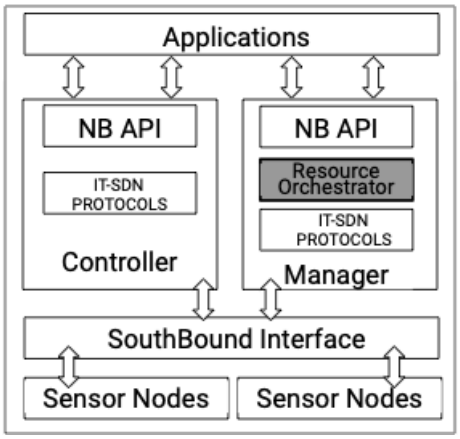


Figure 2: Architecture for IT-SDN Manager with the Resource Orchestrator module

Each node maintains the following metric types:

- accumulated queue delay: the node maintains the information about how long a packet waits in the output queue before it is sent;
- number of data packets the node sent;
- energy available in the node;
- number of lost packets: packets that were dropped in the receiving queue or in the transmission to the wireless medium. It is accumulated during a two-minute window and it considers both data and control packets.

Moreover, we designed and implemented the resource orchestrator within the IT-SDN Manager to enable the application requirements configuration and node management. The applications configure the requirements through a REST API available in the Northbound (NB) interface on the manager.

The resource orchestrator is a module that detects when the nodes do not comply with application requirements. The application configures a threshold for each metric type available in the monitoring module. The orchestrator uses this threshold to detect non-compliant nodes. The manager receives the node information and the orchestrator is responsible for verifying if the metric value complies with the application requirement.

As depicted in Figure 3, the orchestrator works together with the monitoring module to obtain the node metrics. However, they have different functionalities. The monitoring module maintains and obtains the node information from the network, while the orchestrator uses the node metrics to detect nodes that do not comply with the application requirements.

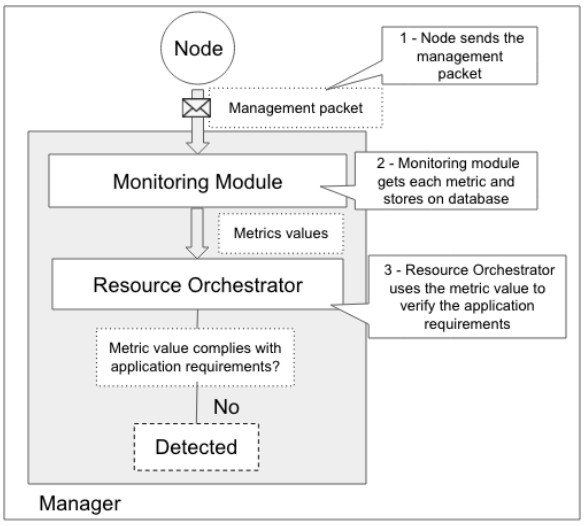


Figure 3: Detection flow: the node sends the management packets and the orchestrator detects when the metric values do not comply with the application requirement

3.1 Implementation

The IT-SDN framework [2] implemented the SDN-controller and provided the software along with the node code[1]. The SDN-controller is composed of two components: (1) a Contiki node that receives packets from the IEEE 802.15.4 network and transmits them to the (2) controller software executed on a PC. The communication between PC and Contiki node is achieved through a serial (USB) connection (or a TCP connection for COOJA simulations). The SDN-controller software executed in a PC uses the neighbor information to maintain the network topology view, and then calculate routes to set up flow entries in the SDN-enabled nodes.

The IT-SDN Manager entity follows the same structure, and its structured is depicted in Figure 4. Thus, we implemented a manager-PC that uses a Contiki node as a bridge between the IEEE 802.15.4 wireless network and the PC. This communication is achieved through the serial connection.

The manager-PC contains two components: (1) the monitoring module and (2) the orchestrator which provides a REST API to enable configurations by the applications. As explained before, the monitoring module receives node metrics that are periodically sent and stores them in a SQLite database. The orchestrator then verifies the nodes that were configured by the applications, comparing the stored metrics with the given

[1] Available at `http://www.larc.usp.br/users/cbmargi/www/it-sdn/`.

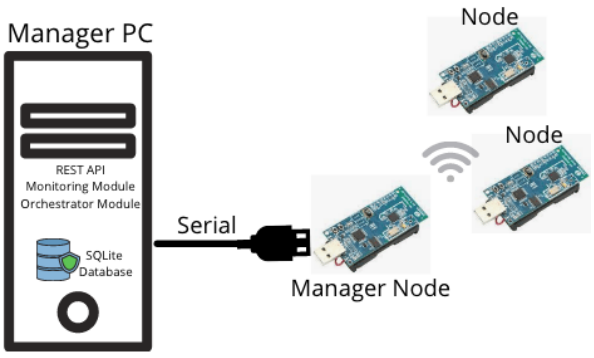


Figure 4: IT-SDN Manager implementation

threshold. Therefore, the orchestrator will be able to detect if a node does not comply with the requirements, and raise an alarm.

4 Experimental Method

We performed the experiments using Cooja, a network simulator and emulator available in Contiki OS [5] emulating Sky Mote devices. We considered a network grid topology with the following sizes: 36, 49, 64, 81 and 100 nodes. Node 1 is the controller in the center grid, node 2 is the data sink and node 3 is the manager. The sink is on the right hand side and the manager is on the left side of the controller. All the other nodes send data packets to the sink. Notice that a node in the middle of the grid has 4 neighbors.

We positioned the controller in the center of the grid, since the IT-SDN performance study [3] showed that it yields better results in terms of delivery rate, delay and energy consumption when compared to the corner position. As we do not aim to study the position impact on this paper, we consider the position with less impact on the mentioned metrics.

Nodes use IEEE 802.15.4 standard as MAC layer and ContikiMAC for radio duty-cycling with the channel check rate set to 16Hz. The IT-SDN [2] [3] framework uses the Dijkstra algorithm for route calculation, ETX as a link metric and a flow table size with 15 entries. All the experiments were executed 10 times with a 60-minute duration each. Results depict the mean value calculated considering a 95% confidence interval. Table 1 summarizes the configuration parameters related to IT-SDN.

Nodes start to send data packets and management packets at a random time of up 3 minutes after the network initialization. The data packet payload has 10 bytes. The data transmission frequency is different for each scenario and application.

We defined five scenarios with different application.

Table 1: Simulation parameters

IT-SDN parameters	
Version	Manager module
Controller retransmission timeout	60 s
ND protocol	Collect-based
Link metric	ETX
Neighbor report max frequency	1 packer per minute
CD protocol	none
Route calculation algorithm	Dijkstra
Route recalculation threshold	20%
Flow table size	15 entries

These scenarios aim to simulate real applications such as monitoring on Smart Cities or Smart Campus [15]. The data frequency depends on the application requirements and tasks. As depicted in Figure 1, suppose that there is a low data frequency application, such as environmental monitoring. Also, there is a high data frequency application, such as the movement detector to identify and count pedestrians. Assume that, instead of having a set of sensing devices for each application, devices will be shared by these different applications, i.e., the same hardware will run, collect and transmit data for different applications. The scenarios are:

- Scenario 1: this is our baseline (BL). The application base (AB) sends 1 data packet every 2 minutes. The manager is not active.
- Scenario 2: the application base (AB) runs along with the manager and orchestrator active. Since the orchestrator is active, it monitors the energy level and packet loss threshold.
- Scenario 3: two applications running - AB and Application 1 (A1). A1 sends 1 data packet per minute, and starts to send data 10 minutes after AB starts. The manager and orchestrator are active, monitoring the energy level and packet loss threshold.
- Scenario 4: two applications running - AB and Application 2 (A2), which sends data every 30 seconds and starts 15 minutes after AB. The manager and orchestrator are active, monitoring the energy level and packet loss threshold.
- Scenario 5: three applications running - AB, A1 and A2, i.e., all the applications run concurrently. The manager and orchestrator are active, monitoring the energy level and packet loss threshold.

Table 2 summarizes the scenarios used in these experiments.

5 THRESHOLD SELECTION

The orchestrator detection depends on the metric monitoring and on the threshold configured by the application. During the application deployment, the threshold is defined through the API manager. The energy level threshold should be related to the expected node lifetime. Nodes begin the simulation with full batteries, thus having 100% available energy. If a new application is deployed when nodes have 20% available energy, the network will have little time to serve the application. On the other hand, if it is deployed when nodes have 90% available energy, the application will last longer.

However, packet loss depends on the network conditions, i.e., application data being transmitted, control overhead, network size, buffer size, and so on. Thus, it is not trivial to determine such threshold.

In order to understand the packet loss threshold that best suits the scenario, we analyze the packet loss behavior according to the application tasks and network sizes. To do so, we executed simulations for scenarios 2, 3, 4 and 5 and several network sizes, as described in Section 4. Then we analyzed the simulation trace files in order to understand the packet loss behavior.

We calculated the mean packet loss for each node in the 10 simulations. For each node we count how many packets were lost and calculate the mean value in each scenario and network size. We depict the results as the heat-map of the grid topology to explore how the node position related to sink, manager or controller, relates to the packet loss. Table 3 shows the heat-map of the mean packet loss for each node according to the scenario and network size.

Each point on the heat-map is one node in its position in the grid topology. Red squares are Sink, Controller and Manager. A black square indicates that no packet is lost, while light gray and/or white squares means more packets are lost.

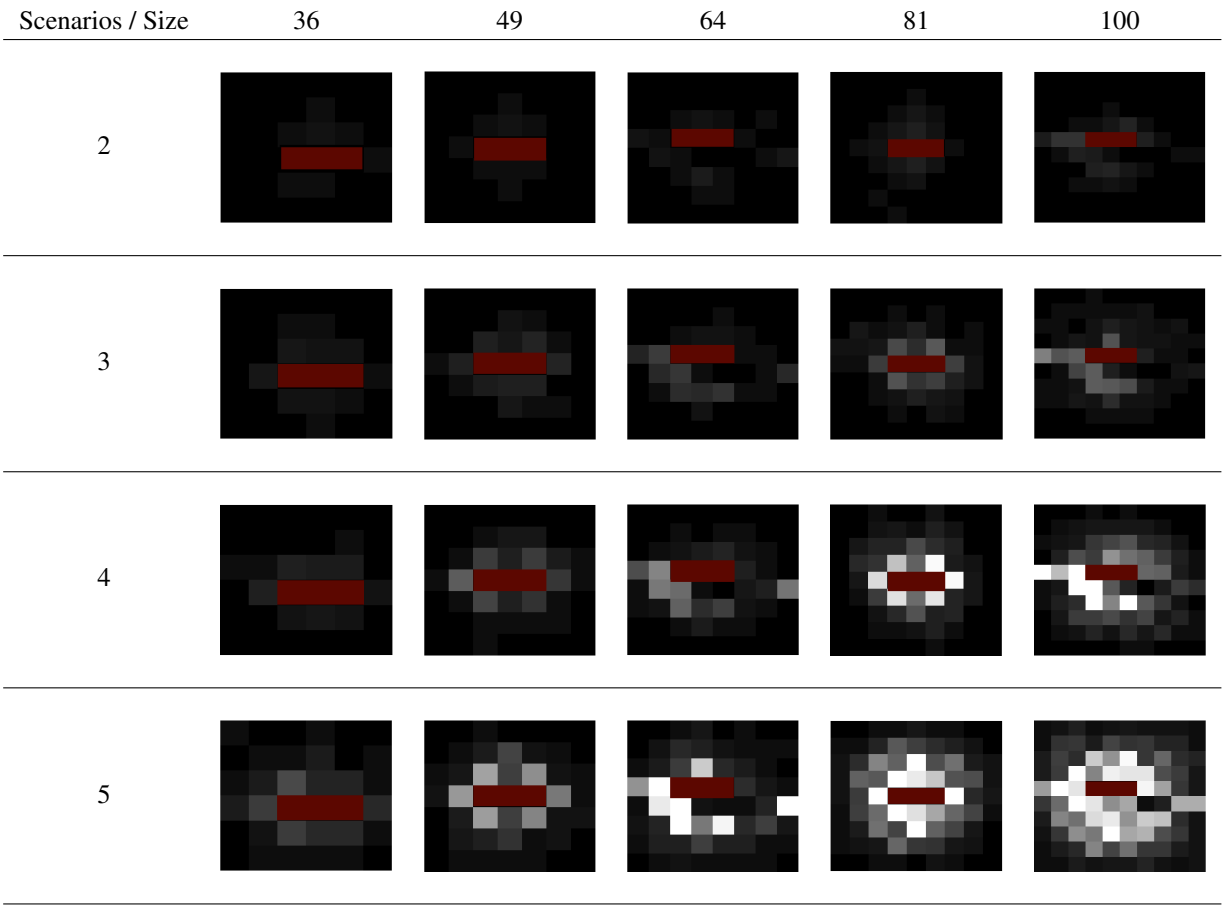
Note that nodes 1-hop from the controller, manager or sink lose more packets when compared to nodes that are far away. We hence use one 1-hop node as the baseline to determine the threshold. We calculate the mean value for this node for each scenario in a two-minute sliding window (SW). Table 4 presents the mean of the SW and also the mean of the total packet loss for each scenario and network size.

In scenario 2, where there is a low data frequency, the packet loss is less than 1 for all the network sizes. Concerning the 36-node network size, packet

Table 2: Scenarios and applications with different data frequency

Scenario	Application	Manager Active	Data packet frequency
1	AB	No	1 every 2 minutes
2	AB	Yes	1 every 2 minutes
3	AB + A1	Yes	1 every 2 minutes (AB) + 1 every minute (A1)
4	AB + A2	Yes	1 every 2 minutes (AB) + 1 every 30 seconds (A2)
5	AB + A1 + A2	Yes	1 every 2 minutes (AB) + 1 every minute (A1) + 1 every 30 seconds (A2)

Table 3: Heat-map for mean packet loss for each node in its grid position in the scenarios and network sizes. Red squares are sink, controller and manager. Black square indicates no packet loss.



loss is greater than one only in scenario 5, where all applications are running. When observing 49 and 64 network sizes, packet loss is less than one for scenarios 2 and 3, as well. For larger network sizes and scenarios with more applications running (i.e., higher data frequency), Table 4 shows that the packet loss increases, as expected.

The orchestrator threshold is used to detect when a node does not comply with the application requirements configured. Therefore, we refer to Table 4 and use the rounded mean as the orchestrator threshold to packet loss. Thus, the threshold is set as 1 for the values smaller than 1 in the table. This approach enables a realistic detection, since it considers the difference in scenarios and network sizes.

6 Results and Discussion

We evaluate the impact of the manager overhead and the orchestrator detection performance. The overhead considers the impact caused by management packets. The performance regards event detection concerning the

Table 4: Mean for two-minute window and total mean loss for 1-hop node (Scenarios x Network size)

		36	49	64	81	100
	Window	0.04	0.16	0.2	0.5	0.9
	Total	4	6	25	30	24
	Window	0.08	0.5	0.7	2.3	3.4
	Total	25	24	48	94	113
	Window	0.5	1.5	2.3	5.4	5.9
	Total	35	107	144	322	545
	Window	2.4	4.2.	7.4	6.8	6.8
	Total	76	213	387	480	520

application requirements.

6.1 Manager Overhead

Concerning the manager overhead, we evaluate the following network metrics: data delivery rate, data delay, energy spent and control overhead.

Figure 5 shows the mean data delivery rate (DR). When comparing scenarios 1 and 2, which have the same application running but the manager is active only in scenario 2. The difference is less than 2%.

Comparing scenarios 2 and 3, where there is one more application running, the difference is at most 3% for 100-node size. In scenario 4, the DR is greater than 95% for network sizes up to 64 nodes, while being about 85% with 100 nodes. The data delivery rate difference in scenario 4 occurs because there is an application sending one data packet every 30 seconds, increasing the number of packets in the network. Since nodes have a limited queue buffer size, more packets in the network would fill the space faster.

For network sizes of up to 64 nodes, the DR is greater than 90%. However, for network sizes with 81 and 100 nodes, the DR is about 82% and 70%, respectively. Note that in scenario 5, all the applications run at the same time with a high data frequency. The DR is expected to drop for the 100-node network size, because there are more nodes sending packets for multiple applications.

We calculated the energy consumption for all the nodes in each scenario to verify the manager impact, which is depicted in Figure 6. Energy consumption in scenario 2 is about 10% greater than our baseline (scenario 1). Since there are more packets in the network, more energy is consumed. As expected, the energy consumption increases with the data frequency and network size for all the scenarios.

Data delay is presented in Figure 7. Scenario 2 presents about 2% of difference when compared to the baseline (scenario 1). For network size of up to 49 nodes, for all the scenarios, the delay increases about 2%. Thus,

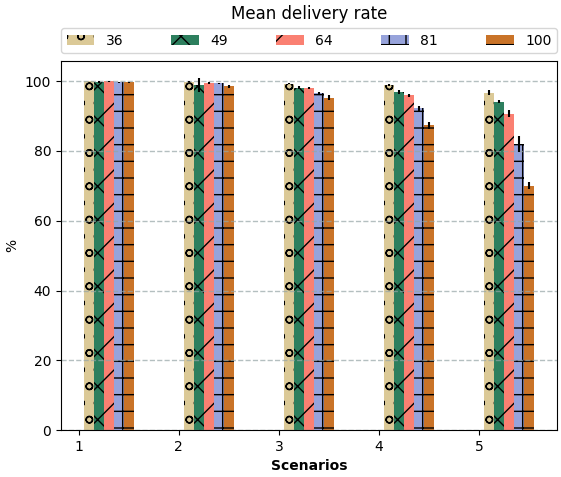


Figure 5: Mean data delivery rate (%) for the scenarios and network size

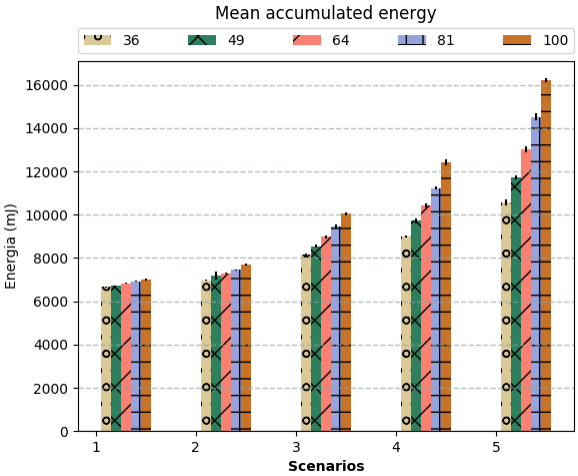


Figure 6: Mean energy consumption (mJ) for each scenario

for small networks even with more applications running at the same time, there is no significant impact on data delay. When the network size increases, for 64, 81 and 100 nodes, in scenario 5, where all the applications are running, the delay is more than doubles when compared to scenario 2. However, note that there are more hops until the packet reaches the controller, manager and sink.

Figure 8 presents the control overhead. The control overhead has a similar behavior to the energy consumption. It increases with the data frequency and network size. However, there is almost twice the number of packets in scenario 2 for network size with 100 nodes when compared to scenario 1. This is expected since the orchestrator and manager are active in scenario 2 and therefore there are management packets.

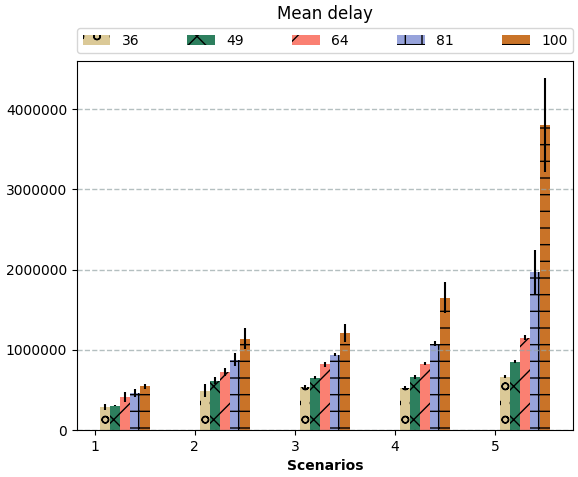


Figure 7: Mean data delay (milliseconds) for each scenario

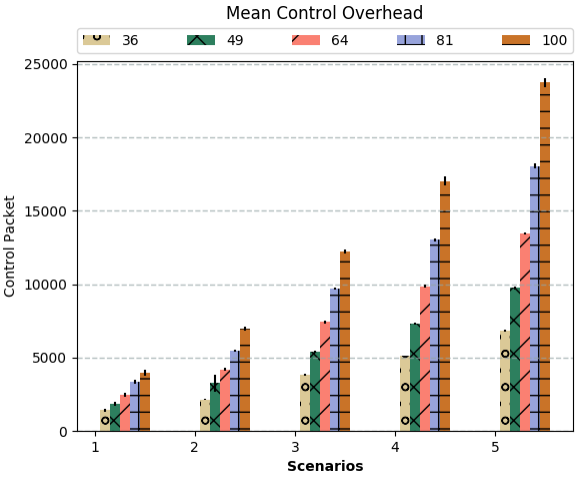


Figure 8: Mean control overhead for each scenario

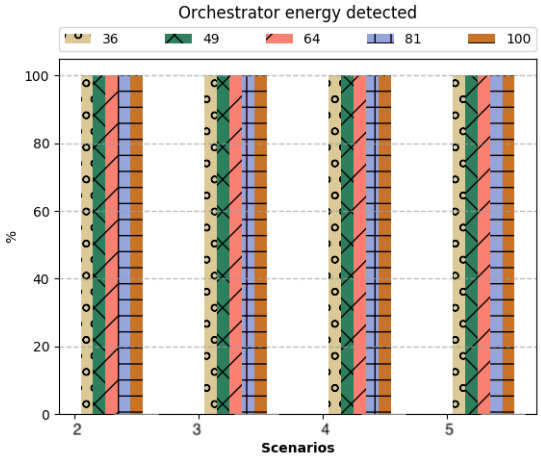


Figure 9: Detection rate for energy level by the Orchestrator

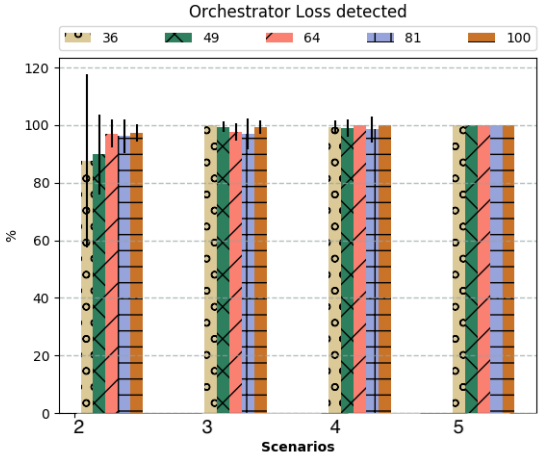


Figure 10: Orchestrator detection rate for packet loss

Although the overhead increases with the network size and application, it does not significantly affect the DR.

In summary, results showed that energy consumption, delay and control overhead increase with applications and network size. The worst scenario is scenario 5, where all the applications run and there are more data packets. However, when we compare scenarios 1 and 2, we notice that the difference in DR is about 2%. Thus, the manager activation does not significantly affect the network metrics here presented.

6.2 Orchestrator Detection Performance

We analyzed the energy and packet loss detection in the orchestrator and compared it with the trace file from Cooja. Scenario 1 is not depicted in the graphs, since the manager and the orchestrator are disabled.

Figure 9 shows the detection for energy level. The orchestrator detected 100% of the cases. So, every node that spent 10% of the energy is detected. The orchestrator detects all the cases because the energy variation has no significant changes in the period of one minute. Hence, the node sends the same energy information in multiple packets. Even if one of these packets is lost, the manager receives the energy information and the orchestrator detects it.

Figure 10 shows the mean detection rate for packet loss. If the same node is detected more than once, only the first one is considered. Besides, we only consider the lost packets after the node starts to send management packet.

The detection rate is greater than 97% for all the cases, except for scenario 2, where we have the worst detection

for the 36 and 49 network sizes. We analyzed the simulations for these network sizes to better understand why they presented such an error. For the 36-node network size, one simulation (out of ten) detected about 75% and the other simulation did not detect any of the nodes. For the 49-node network size, two simulations (out of ten) detected 66% and the remaining simulations detected 90% of the nodes.

However, Table 4 allows observing that for scenario 2, the networks with 36 and 49 nodes have a mean packet loss of 4 and 6 packets. The error increases when few packets are lost and only one node is not detected. Yet, these network sizes have just a few cases of packet loss, which did not significantly impact the network. Moreover, the detection rate increases with network size and data frequency, because more events are present and the node could be detected more than once.

The results showed that the orchestrator is able to detect when a node reaches the configured threshold. The threshold is configured according to the application requirements, but it must also take into account the network size and load.

7 Conclusions

In order to improve the coverage, reduce costs, and make better use of the available resources, sharing the infrastructure among multiple applications is necessary in very large IoTs. However, multiple applications running simultaneously on the same WSN/IoT could lead to network overload or failure to meet application requirements. Therefore, management and orchestration are of critical importance to share the resources without overloading the sensor nodes and network capacity. Related works include approaches that verify the node and network state before admitting new applications, but these works do not continuously monitor nodes and network in order to detect changes that could negatively impact the performance.

To address such a gap, we designed and evaluated the IT-SDN Manager, a resource orchestrator that monitors network metrics and detects when nodes that do not comply with predefined application requirements. It monitors the network metrics to identify nodes that reach a certain threshold of energy available and packet loss. The threshold is defined according to the network size and applications executed on the network.

Results show that the manager does not introduce a significant overhead, since it does not considerably change the data delivery rate, the node energy or the data delay. Concerning the orchestrator detection performance, it detects all the nodes that reach the available energy threshold, and at least 85% of the nodes that reach the packet loss threshold with an average of 97%. Packet loss detection increases with the network size and data frequency.

As future work, we aim to use the orchestrator detection to trigger routing changes and/or task relocation to improve resource utilization and resource sharing. Furthermore, we aim increase the network size for the simulation topology in order to improve the scalability study.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and by the ELIOT ANR-18-CE40-0030 and FAPESP 2018/12579-7 project.

References

[1] S. Akkermans, W. Daniels, G. Sankar R., B. Crispo, and D. Hughes, "CerberOS: A Resource-Secure OS for Sharing IoT Devices," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '17. USA: Junction Publishing, 2017, p. 96–107.

[2] R. C. A. Alves, D. Oliveira, G. Núñez, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," in *XXXV Simpósio Brasileiro de Redes de Computadores - SBRC 2017*, 2017.

[3] R. C. A. Alves, D. A. G. Oliveira, G. A. Nunez Segura, and C. B. Margi, "The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks," *IEEE Access*, vol. 7, pp. 115 093–115 108, 2019.

[4] A. Chakraborty, S. Misra, A. Mondal, and M. S. Obaidat, "SensOrch: QoS-Aware Resource Orchestration for Provisioning Sensors-as-a-Service," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[5] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, Nov 2004, pp. 455–462.

[6] Z. Li and A. Zhong, "Resource allocation in wireless powered virtualized sensor networks," *IEEE Access*, vol. 8, pp. 40 327–40 336, 2020.

[7] T. C. Luz, G. A. Nunez, C. B. Margi, and F. L. Verdi, "In-network performance measurements for software defined wireless sensor networks," in *2019 IEEE 16th International Conference on*

Networking, Sensing and Control (ICNSC), 2019, pp. 206–211.

[8] W. Muhamad, N. B. Kurniawan, Suhardi, and S. Yazid, "Smart campus features, technologies, and applications: A systematic literature review," in *2017 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2017, pp. 384–391.

[9] M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke, "SDNMM—A Generic SDN-Based Modular Management System for Wireless Sensor Networks," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2347–2357, 2020.

[10] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software Defined Networking for Improved Wireless Sensor Network Management: A Survey," *Sensors*, vol. 17, no. 5, 2017.

[11] T. M. C. Nguyen, D. B. Hoang, and T. Dat Dang, "Toward a programmable software-defined IoT architecture for sensor service provision on demand," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, 2017, pp. 1–6.

[12] RFC. (2015) Software-Defined Networking (SDN): Layers and Architecture Terminology. [Online]. Available: https://tools.ietf.org/html/rfc7426--Accessed12-2020.

[13] P. Semasinghe, S. Maghsudi, and E. Hossain, "Game Theoretic Mechanisms for Resource Management in Massive Wireless IoT Systems," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 121–127, February 2017.

[14] R. Thupae, B. Isong, N. Gasela, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks mangement and security challenges: A review," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 4736–4741.

[15] H. Zemrane, Y. Baddi, and A. Hasbi, "SDN-Based Solutions to Improve IOT: Survey," in *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, 2018, pp. 588–593.

AUTHOR BIOGRAPHIES

Thamires de Campos Luz is a Ph.D. student at the Universidade de Sao Paulo. She received her MSc. degree (2015) in Computer Science from the Universidade Federal de São Carlos and the B.Sc in Data Processing from Faculdade de Tecnologia de Sorocaba. Her research interests include network management and orchestration in software-defined networking.

Cintia Borges Margi obtained her Ph.D. in Computer Engineering at the University of California Santa Cruz (2006), and her Habilitation (Livre Docência) (2015) in Computer Networks from the Universidade de São Paulo. She has been an Associate Professor in the Computer and Digital Systems Engineering department at Escola Politecnica – Universidade de São Paulo (EPUSP) since 2015, where she started as an Assistant Professor in 2010. In 2007-2010 she was an Assistant Professor at Escola de Artes, Ciencias e Humanidades da Universidade de São Paulo (EACH-USP). Her research interests include: wireless sensor networks and software-defined networking.

Fábio Luciano Verdi is an Associate Professor at the Computing Department in the Federal University of São Carlos (UFSCar), campus Sorocaba. He received his Master degree in Computer Science and Ph.D. degree in Electrical Engineering both from the State University of Campinas (UNICAMP). Fábio has been working with data centers, cloud computing and SDN. He is the coordinator of the LERIS Research Group and has been leading projects in the area of monitoring of virtual resources and cloud infrastructures.