In-network performance measurements for Software Defined Wireless Sensor Networks

Thamires C. Luz
Universidade de São Paulo
São Paulo - SP, Brazil
thamires.luz@usp.br

Gustavo A. Nunez
Universidade de São Paulo
São Paulo - SP, Brazil
gustavoalonso.nunez@usp.br

Cíntia B. Margi
Universidade de São Paulo
São Paulo - SP, Brazil
cintia@usp.br

Fábio L. Verdi
Universidade Federal de São Carlos
Sorocaba - SP, Brazil
verdi@ufscar.br

***Abstract*—Wireless Sensor Networks (WSN) are key enablers for several applications such as smart agriculture and smart cities. Software Defined Wireless Sensor Networking (SDWSN) paradigm improves WSN resource sharing, flexibility, and management. Monitoring network performance is essential to network management and it also plays an important role in making better use of scarce resources. However, obtaining network performance metrics in WSN is not trivial due to energy and computational constraints on the WSN nodes. The main contribution of this paper is a performance monitoring module for IT-SDN, an SDWSN framework. This module collects information about the data packets transmitted, queue delay and available energy on the WSN nodes, and sends the information as requested by the controller. The results showed that the monitoring module neither impacts the delivery rate, delay and overhead of the network, nor the energy consumption of the node. Also, the data collected by the monitoring module is pretty similar to the real values.**

***Index Terms*—network monitoring, software defined wireless sensor networks, management**

I. Introduction

A Wireless Sensor Network (WSN) has a key role to enable the use of IoT (Internet of Things) devices [1], giving raise to several applications, such as smart agriculture and smart cities. By using a WSN infrastructure, sensor nodes are capable of sending information through multiple hops until it reaches the data collection point. The main WSN challenge is resource limitation on the nodes (i.e., constrained processor, memory and energy source).

Software-Defined Networking (SDN) [2] is considered one solution that can improve the use of resources on WSN nodes. The SDN paradigm separates control and data planes, where the control plane is maintained by a separate entity called controller, which has the global view of the network topology [2]. The fusion of SDN and WSN generated the Software-Defined Wireless Sensor Networks (SDWSN) paradigm.

With the SDWSN, network management can be done dynamically and programmed by software. Network management depends on how the network is monitored and what information is collected to take actions on the network [3]. These actions are generally taken based on measurements and network metrics such as packet routing, link loss ratio, route delay and delivery rate. However, monitoring WSN performance measurements is not a trivial task.

Several management proposals present a monitoring system that collects selected information locally and sends it constantly to a repository. While this approach keeps the repository updated, sending this information frequently increases traffic and might cause network congestion, thus decreasing data delivery rate [4]. For example, Sympathy [5] is a tool to detect failures on the nodes that actively collect information related to routing table and flow information.

Another approach is to monitor and store the performance information in the device. When necessary, the entity in charge of network monitoring requests the information. Therefore, there is a trade-off between network traffic and local storage. SNMP for 6LowPan [6] uses this approach; however, headers compression and MIB storage may not suit all types of nodes.

These two approaches depict a trade-off. Constantly generating packets to send the performance information imposes high traffic load on WSN, which is very costly due to resource limitations and is likely to exhaust the energy available in the node [4]. Conversely, storing information in the node may also overflow the node capacity.

To address the challenge of the traffic load and node capacity, as both approaches have their trade-offs, we aim to reduce the traffic flow by sending information only when requested, and also save information in the node memory. It differs from the 6LowPAN approach, which uses MIB storage and may not suit all types of nodes capacity

The main contribution of this paper is the design, implementation and evaluation of a monitoring module within a SDWSN framework, IT-SDN [7]. This monitoring module provides network and node metrics, enabling real-time network monitoring. More specifically, it collects three performance metrics: node energy, number of data packets sent by the node, and the accumulated delay of sent data packets. We evaluate its overhead and compare the collected information obtained from a simulator.

We extend IT-SDN to include monitoring messages that will be exchanged between controller and nodes. We specify the parameters to be monitored and how they are to be collected, aiming to be the least intrusive possible. Results show that the monitoring module does not cause a large overhead and it does not impact the delivery rate, delay or energy consumption.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. C. B. Margi is supported by CNPq research fellowship #307304/2015-9

978-1-7281-0084- 5/19/$31.00 ©2019 IEEE

II. IT-SDN MONITORING MODULE

The IT-SDN monitoring module was implemented in Contiki OS [8]. The monitoring module assumes that the node maintains its own information locally and sends it according to the controller request defined by the WSN administrator. It is not the scope of this paper to determine when and which information the controller decides to request from the nodes. We focus on the monitoring task and how the information is obtained by the SDN controller.

A. Performance information

In the monitoring module, we collect three metrics: accumulated queue delay in the node, number of data packets the nodes sent and the energy of the node.

Queue delay: The end-to-end route delay metric can be expressed as the difference between the time the packet was sent and the time it was received. Besides, it can be calculated by the sum of the transmission delay, propagation delay and queue delay. In WSN, it is not simple to calculate these delays, since nodes have simple hardware usually without synchronized clocks. One approach is to synchronize the nodes, and then calculate the end-to-end route delay [9] [10]. However, node synchronization requires more actions to be taken and more packets in the network [11]. Another approach is to use the individual time of each node and sum these times [12]. If the medium propagation delay and the number of bytes sent are known, then each node has to calculate its queue delay. Thus, the route delay can be given by the sum of the queue delay for each node in the route, the propagation delay and the transmission delay.

We followed the second approach in our monitoring module. Each node calculates the accumulated queue delay considering two situations: sending and forwarding packets. For sending packets, we calculate the difference between the current local timestamp when a data packet comes in the transmission queue and when the packet is sent. For forwarding packets, the accumulated queue delay considers the difference between local timestamp in receiving queue and when a packet is sent.

Number of data packets sent: The delivery rate and packet loss are important network metrics. The delivery rate is calculated by the ratio of data received and the data sent. Packet loss is calculated by the difference between the packet sent and the packet received. Hence, the quantity of data packets the nodes sent can be used to calculate packet loss and delivery rate. In this monitoring module, the node keeps a counter that is incremented every time the node sends a data packet. When the controller requests this counter, it should also request how much data packets were received at the sink from that particular node in order to calculate the delivery rate.

Energy available: Energy is crucial for the WSN nodes. To obtain the node's energy available, we used Energest [13]. This module calculates the accumulated energy consumption for a node each minute and how it impacts on the node energy. When the controller requests the energy for a node, it will be returned the amount of energy available relative to the initial value.

B. Information storage

The performance information could be stored in RAM or in a text file in ROM. RAM is volatile and one must be careful not to overflow its capacity. On the other hand, storing data in a text file, such as a MIB, requires a file system, such as Coffee File System (CFS) [14]. Yet the amount of ROM necessary to support the CFS, IT-SDN and the application would not fit the ROM in a typical device. Therefore, we decided to store the performance information in RAM.

Number of data packets sent information has 1 byte, and the *queue delay* has 4 bytes. Since both values are accumulated, we need to reset them in order to keep the information fresh and to avoid memory overflow. Our approach is to reset both according to a predetermined number of data packets sent, which is defined in the configuration file.

C. Monitoring message exchange

To implement the communication between the controller and the node, two IT-SDN packets were created: Request Packet and Monitoring Packet.

The Request Packet is routed by the source and is composed by the IT-SDN header (6 bytes), the next hop address (2 bytes), the destination (2 bytes), the path length (1 byte), and the message (2 bytes). The message indicates to the node (destination) which performance metrics the controller requested.

Each bit of the message works as a flag of a specific metric, whereby: if the bit is set to '1', it was requested; and if the bit is set to '0', it was not. Thus, 2 bytes can represent 16 different metrics. The total frame is composed by the Request Packet plus the MAC header (9 bytes) and the FCS (2 bytes). This means a total of 24 bytes out of 127 bytes supported by the IEEE 802.15.4 standard.

Differently from the Request Packet, the Monitoring Packet is routed using a flow identification (SDN_CONTROLLER_FLOW). The monitoring packet is composed of the IT-SDN header (6 bytes), the flow ID (2 bytes), the performance metrics sent flags (2 bytes), and the metrics information (up to 64 bytes). Each metric is composed only by its value (4 bytes). The total frame is composed by the Monitoring Packet plus the MAC header (9 bytes) and the FCS (2 bytes). It means a maximum packet size of 85 bytes.

III. EXPERIMENTAL METHOD

In order to evaluate our proposal, we analyze the overhead and also the accuracy of the information obtained by the monitoring module. We consider aspects of the frequency that controller requests the monitoring packets and when the information is reset in the node.

We performed the experiments using Cooja, a network simulator of emulated Contiki WSN nodes [15]. We considered a network grid topology with several sizes, where node 1 was set as the controller, node 2 as the sink, and all other nodes sending data packets to the sink. Nodes start to send data packets at a random time up to 3 minutes after the network

Table I
SIMULATION PARAMETERS

Simulation parameters	
Topology	Square grid
Number of nodes	25, 36, 49, 64, 81
Node boot interval	$[0, 1]$ s
Number of sinks	1
Data traffic rate	1 per minute
Data payload size	10 bytes
Data traffic start time	$[2, 3]$ min
Running time for simulation	1 hour
ContikiMAC channel check rate	16 Hz
MAC layer	CSMA
Radio Medium Model	UDGM
Device	Sky mote
IT-SDN parameters	
Version	0.4
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

initialization. The data packet payload has 10 bytes. Table I summarizes the configuration parameters related to IT-SDN and Contiki OS.

We need to determine when the accumulated information is reset. We chose to reset the accumulated information at a 50-minutes interval (50 packets). In order to understand the impact of the reset interval, we decided to evaluate a different reset value, choosing to reset every 100 packets as well.

Moreover, we consider the controller requests frequency. For each controller request, two packets will be transmitted: one packet for the request and one for the response. To evaluate the impact of these new packets and also the accuracy of the data obtained by the monitoring module, the controller requests the information for one node every minute. This means that, in the first minute, the controller requests the information in node 3; in the second minute, the information in node 4 and successively, until it reaches all the nodes and starts again. Moreover, we consider the controller request made for 50% of the nodes. With this, we can compare if with fewer controller requests, the information can be representative for all the network nodes.

Furthermore, to compare the overhead of this monitoring module, we considered IT-SDN running without the monitoring module (our baseline) and then running with the monitoring module. Each simulation was performed 10 times for one hour.

Table II summarizes the scenarios described. When presenting our results (Section IV), we refer to the first column of this table.

Table II
SCENARIOS ANALYZED FOR ALL THE NETWORK SIZES.

	Reset information	Controller Request	Monit.
1	–	–	No
2	50	All nodes	Yes
3	50	50%	Yes
4	100	All nodes	Yes
5	100	50%	Yes

We compared the results from the monitoring module with the simulation trace files from Cooja to evaluate its accuracy. We also used the traces to calculate the mean delay, total energy spent, mean delivery rate and the overhead of control packets when the monitoring module is active.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

This section presents results and related discussion considering two aspects: the impact of the monitoring module on the WSN performance, and the accuracy of the information obtained by the monitoring module. The results were calculated considering a 95% confidence interval for the mean for all the 10 simulation runs for each scenario (as shown in Table II).

A. Monitoring module performance

In order to understand the impact of the monitoring module, we evaluate the WSN performance regarding delivery rate, delay, energy spent and control overhead. The delivery rate and the delay are calculated just for the data packets.

Figure 1 shows the mean data delivery rate for all the 10 simulation runs for each scenario. In most of the network sizes, the delivery rate for the scenarios with the monitoring module active is very close to our baseline, scenario 1 (without monitoring). In the worst case, the difference is less than 3%. Thus, the delivery rate with the monitoring module active in the network is about the same as that in our baseline.

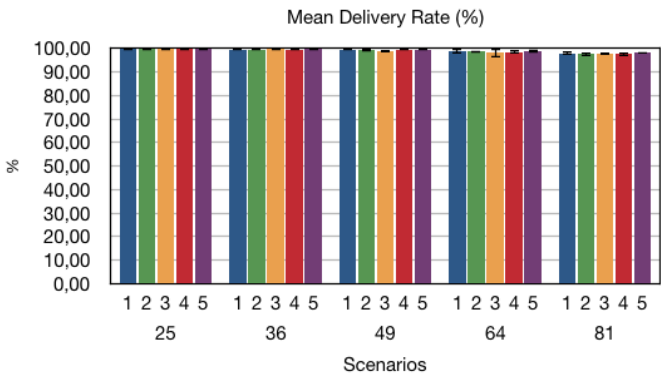


Figure 1. Mean delivery rate (%) for 10 simulation runs for each scenario.

Another important metric we evaluate is the data mean delay, depicted in Figure 2. We observe that the delay in

most of the scenarios with the monitoring module active is similar to scenario 1 (our baseline), with an about 200-millisecond difference. The exceptions are scenarios 3 and 5 for the network with 81 nodes, where the difference is up to 1200 milliseconds. The controller monitoring requests for these scenarios target only the odd nodes in order to achieve 50% of the network. This pattern causes requests to nodes that are more hops away from the sink, thus increasing the delay. We also observe that the delay increases for networks with 64 and 81 nodes. However, such delay also increases in our baseline scenario (without monitoring) since there are more packets in the network. Thus, we conclude that the monitoring module activation does not impact the network delay.

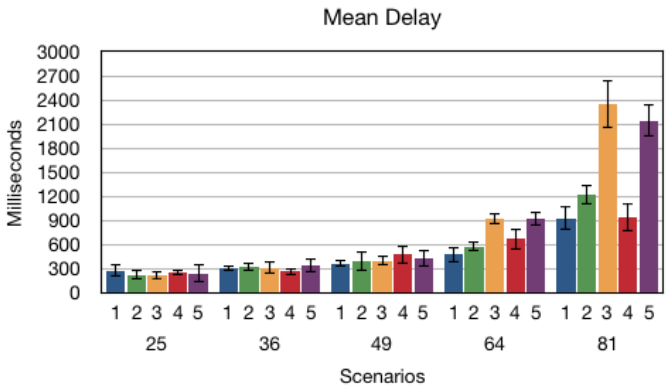


Figure 2. Mean delay (milliseconds) for 10 simulations for each scenario.

We calculated the energy consumption for all the nodes in each scenario and the network size to verify the difference with the monitoring module active. The energy consumption spent is shown in Figure 3. Note that the mean energy consumption spent is very similar in all the scenarios, being no more than 2% when compared to scenario 1. The energy consumption spent increases when there are more nodes in the network for all the scenarios, including scenario 1 (our baseline). Therefore, we conclude that the monitoring module does not impact the energy consumption on the nodes.

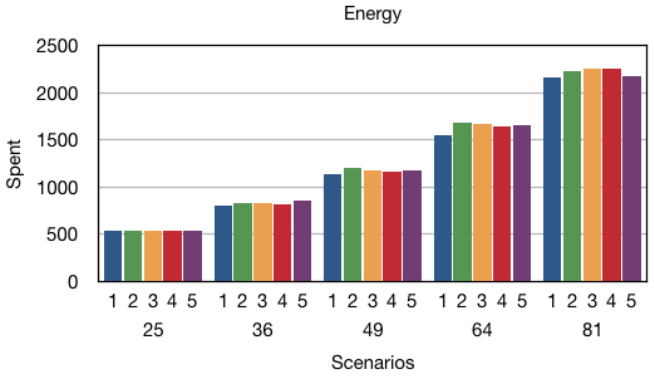


Figure 3. Mean energy consumption for all the nodes in each scenario.

We calculate the control overhead for all IT-SDN control packets. Figure 4 shows the sum of all the control packets in the experiments. Notice that the control overhead is very similar in all the scenarios and the difference is about 10% when compared to scenario 1 (without monitoring module). For the topologies with 25 and 36 nodes, the overhead in scenario 1 is greater than the other scenarios. This may occur because of some congestion or retransmission. In the worst case, for the network with 81 nodes, the difference is about 22% when compared with scenario 1. However, we consider this difference acceptable since there are more nodes receiving requests from the controller and these packets do not impact the metrics discussed earlier. We thus conclude that the new control packets from the monitoring module do not cause a significant overhead that impacts the other metrics with this controller request frequency.

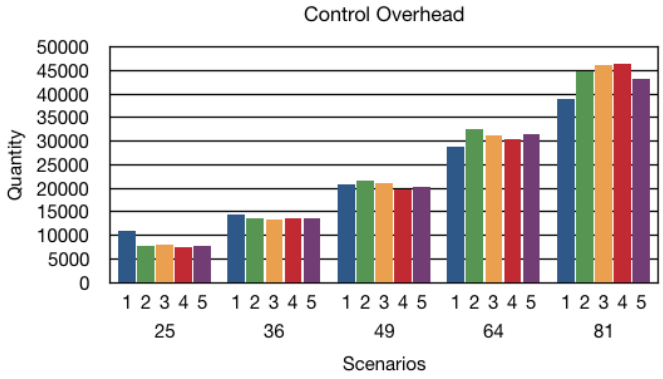


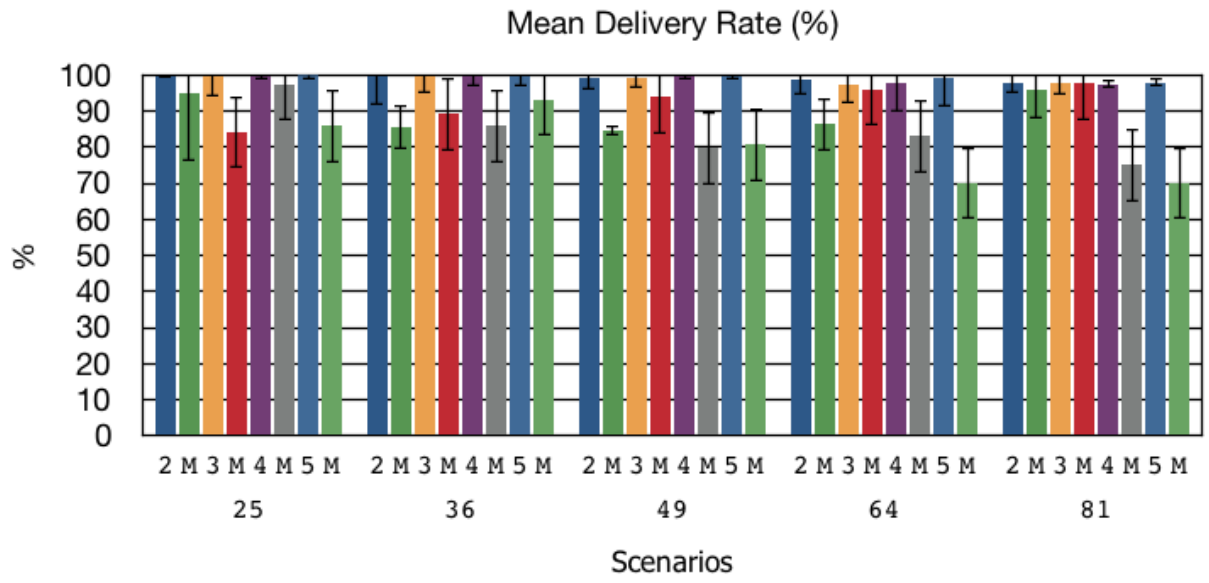
Figure 4. Sum of all control packets of IT-SDN for each scenario.

B. *Monitoring module accuracy*

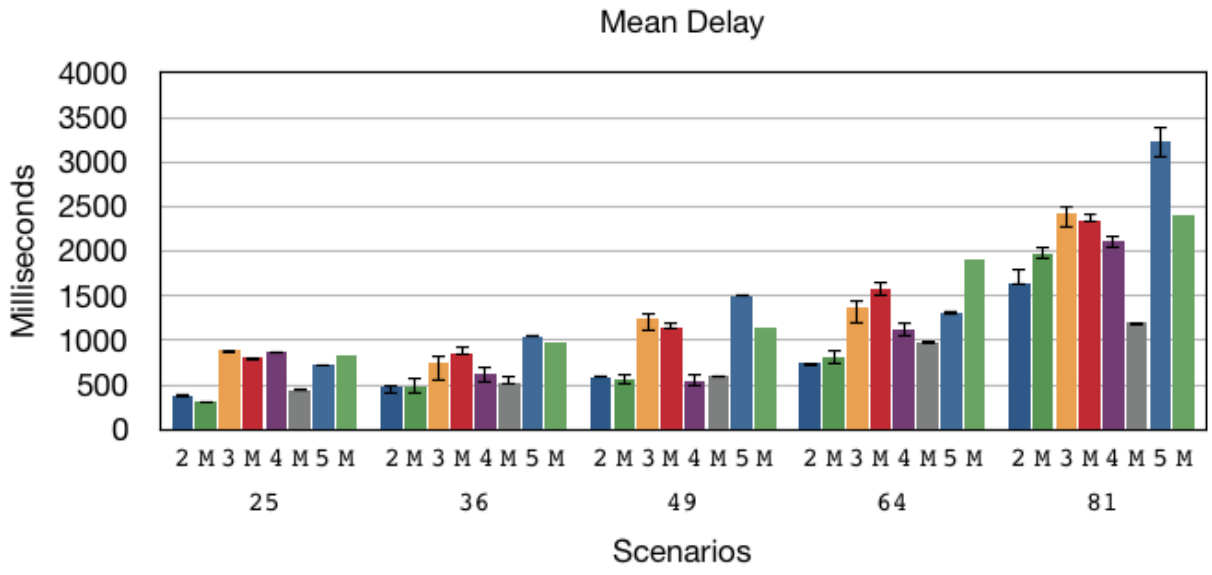
To evaluate the monitoring module accuracy, we compare the values obtained by the monitoring module with the values obtained by the Cooja trace files. We consider delivery rate and delay as the main metrics to be compared. We only consider data packets to compare these metrics.

Figure 5(a) shows that for most of the scenarios, the delivery rate obtained by the monitoring module presented about 10% difference when compared to the data obtained in trace files from Cooja. We notice that for scenarios 3 and 5 the data obtained by the monitoring module presents about 15% difference when compared to the value obtained by the trace file from Cooja. This indicates that the controller request for 50% of the nodes lost some information. However, as we see in other cases, as in scenario 2 with 36, 49 and 64 network sizes, the data obtained by the monitoring module is about 15% different from the data obtained by Cooja. Hence, we conclude that the controller request for 50% of the nodes every minute for one node can represent the general nodes even presenting a large difference from the scenarios with controller request to all nodes.

In general, the delay calculated by the monitoring module presented a difference of about 500 milliseconds when compared to the values obtained by the trace files from Cooja



(a) Delivery rate



(b) Delay

Figure 5. Results of delivery rate and delay for the nodes requested by the controller and the data obtained by the monitoring module(M).

(Figure 5(b)). Except for scenarios 3 and 5, we see a greater difference for networks with more than 49 nodes (about 1.000 milliseconds). Also, in scenario 5, that the reset information occurs every 100 packets, we see that for a network with 64 nodes, the monitoring module obtained a value greater than that obtained by trace file. From this, we conclude that the information obtained for 50% of the nodes can represent the overall of the network, but the information reset should be conducted with fewer packets to obtain fresh data. Also, a controller request must be considered more frequently or for more nodes at the same time to verify the data obtained.

V. Related Work

The main work related to metric monitoring is TinySDM [3]. TinySDM is a software-defined measurement architecture for WSN that allows metrics to be collected dynamically. The evaluation of TinySDM was based on the comparison of tree measure tools: path, delay and data collecting information. However, the implementation was for TinyOS, which limits its use and is not currently updated.

A framework for performance monitoring was proposed and it aims to provide a global definition for metric monitoring [16]. The authors define generic metrics based on an industrial project to generalize which metric should be

collected for efficient monitoring. The metrics are divided into seven categories: delay tolerance, loss tolerance, capacity of the link, reliability, energy efficiency, criticality and fault tolerance. The main goal to calculate the metrics is to infer most of the metrics from the traffic flow and to calculate what is possible in the node to avoid unnecessary information traffic. In our work, we collect three metrics that can fit the seven categories mentioned by these authors.

Lindh and Orhan [17] presented a measurement-based performance management system for WSN that analyzes some metrics for new requests to join the network. A performance meter is implemented in the sensor nodes that have two counters: the number of received and sent packets; and a counter of bytes. These counters measurements are set in a monitoring packet, which is sent to a coordinator node after a set of data packet blocks. It was also implemented for TinyOS. Our work follows the same idea of collecting measurement information from the node and sending it to be processed in a management system. However, our work differs in three aspects: first, we present the measurement collection in the SDWSN context; second, our monitoring packets are just sent to the source when requested, reducing the number of packets in the network; and third: the implementation with IT-SDN is not limited to TinyOS.

The works cited in this section propose different solutions for obtaining the metrics. Yet, most of them have limitations due to their operating system. Our work implements a solution for metric monitoring that can be used in any management system that connects with the controller in a SDWSN context. The implementation uses IT-SDN that is available for download. The results show that the monitoring module activation obtains data important for network monitoring and management, and it does not impact the main metrics: data delivery rate, network delay and energy consumption of the nodes.

VI. Conclusion

The metric monitoring task is crucial to maintain and to manage the network. However, it is not a trivial task in the Software Defined Wireless Sensor Networks context.

In the IT-SDN monitoring module proposed here, we presented a solution to obtain some metrics (quantity of data packets sent by the node, node energy and delay of the node), which can be used to take actions on the network. It was implemented to maintain the monitoring information on each node and only send it when requested by the controller.

By using the metrics monitored by this module, a management system could calculate the delivery rate, route delay and the energy level of the nodes to take actions dynamically and automatized.

The results showed that the monitoring packets do not significantly increase the network control packets. The difference between the scenarios with the monitoring module in the worst case is about 22% when compared to our baseline. However, the data delivery rate, the network delay and the node energy consumption are not impacted by the monitoring module activation. Moreover, it also showed that the data obtained by the monitoring module is pretty similar to the data obtained by the Cooja simulator. Only in scenarios where the information reset occurs every 100 packets, the results are less accurate since the data collected is not fresh.

As future work, we intend to analyze the impact of different controller request frequencies, such as requesting information for more than one node every minute Besides, we plan to analyze other data packet frequencies (1 data packet every 30 seconds or 1 data packet every 5 minutes) and how the information reset should work up with them to present fresh data.

References

[1] C. B. Margi, R. C. A. Alves, and J. Sepulveda, "Sensing as a service: Secure wireless sensor network infrastructure sharing for the internet of things," *Open Journal of Internet of Things (OJIOT)*, vol. 3, no. 1, pp. 91–102, 2017.

[2] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[3] C. Cao, L. Luo, Y. Gao, W. Dong, and C. Chen, "TinySDM: Software defined measurement in wireless sensor networks," in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2016, pp. 1–12.

[4] Y. Liu, K. Liu, and M. Li, "Passive diagnosis for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1132–1144, Aug 2010.

[5] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '05, 2005, pp. 255–267.

[6] J. Shanthini and S. Vijayakumar, "Modified simple network management protocol for 6Lowpan," *Procedia Engineering*, vol. 38, pp. 1024 – 1029, 2012, iNTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING.

[7] 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.

[8] 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.

[9] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," *Netwrk. Mag. of Global Internetwkg.*, vol. 18, no. 4, pp. 45–50, Jul. 2004.

[10] S. G. Ping, "Delay measurement time synchronization for wireless sensor networks," 2003.

[11] Z. B., B. W., and Z. S., "Delay measurement in sensor networks using passive air monitoring," UCONN, Tech. Rep., 2007.

[12] K. Liu, Q. Ma, H. Liu, Z. Cao, and Y. Liu, "End-to-end delay measurement in wireless sensor networks without synchronization," in *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, Oct 2013, pp. 583–591.

[13] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, ser. EmNets '07, 2007, pp. 28–32.

[14] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt, "Enabling large-scale storage in sensor networks with the coffee file system," in *2009 International Conference on Information Processing in Sensor Networks*, pp. 349–360.

[15] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 641–648.

[16] V. Pereira, J. S. Silva, and E. Monteiro, "A framework for wireless sensor networks performance monitoring," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012, pp. 1–7.

[17] T. Lindh and I. Orhan, "Performance monitoring and control in contention-based wireless sensor networks," in *2009 6th International Symposium on Wireless Communication Systems*, Sept 2009, pp. 507–511.