Open Access

Open Journal of Internet of Things (OJIOT)
Volume 4, Issue 1, 2018
http://www.ronpub.com/ojiot
ISSN 2364-7108

Software-Defined Wireless Sensor Networks Approach: Southbound Protocol and Its Performance Evaluation

Cíntia Borges Margi, Renan C. A. Alves, Gustavo A. Nunez Segura, Doriedson A. G. Oliveira

Universidade de São Paulo, Av. Prof. Luciano Gualberto, travessa 3, n.158, São Paulo, Brazil
{cintia, renanalves, gustavoalonso.nunez, doriedson}@usp.br

Abstract

Software Defined Networking (SDN) has been identified as a promising network paradigm for Wireless Sensor Networks (WSN) and the Internet of Things. It is a key tool for enabling Sensing as a Service, which provides infrastructure sharing thus reducing operational costs. While a few proposals on SDN southbound protocols designed for WSN are found in the literature, they lack adequate performance analysis. In this paper, we review IT-SDN main features and present a performance evaluation with all the sensing nodes transmitting data periodically. We conducted a number of experiments varying the number of nodes and assessing the impact of flow table maximum capacity. We assessed the metrics of data delivery, data delay, control overhead and energy consumption in order to show the tradeoffs of using IT-SDN in comparison to the IETF RPL routing protocol. We discuss the main challenges still faced by IT-SDN in larger WSN, and how they could be addressed to make IT-SDN use worthwhile.

Type of Paper and Keywords

Regular research paper: *software-defined networking, southbound protocol, performance evaluation*

1 Introduction

Software Defined Networking (SDN) has been identified as a promising network paradigm for Wireless Sensor Networks (WSN) and the Internet of Things (IoT). While WSN is considered by several authors as a key technology for IoT, these scenarios are distinct. A WSN deployment is typically specific purpose with the infrastructure owned by the same entity that is interested in the harvested data. Usually the devices are homogeneous in terms of hardware specification, and the convergecast pattern prevails, with little or no packets flowing from the sink to the sensing nodes.

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2018)* in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil. The proceedings of VLIoT@VLDB 2018 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

On the other hand, the Internet of Things supports a variety of scenarios. Small scale deployments, such as smart homes, present heterogeneity of devices. For example, simple sensors tend to be more resource constrained and operate over energy efficient radio technologies, while more powerful nodes (such as cellphones and electronic appliances) may coexist in the same network. Also, the devices are expected to be remotely configurable, requiring other data patterns than the traditional convergecast.

Large deployments, such as smart cities, also present heterogeneity of devices and diversity of communication patterns. Since they are expensive and complex to deploy, these scenarios would benefit from infrastructure sharing. In this case, the infrastructure owner may be different from the application owner, with multiple applications leveraging the same installed base. Furthermore, the infrastructure could be composed by devices owned by different entities, although management should be orchestrated. The Sensing as a Service architecture covers these features [19]. SDN is an approach to manage networks, which separates the control plane from the data plane. The routing (or forwarding) decisions are made by the SDN controller based on received network information.

As surveyed by Kobo *et al.*, several authors have highlighted how SDN would bring flexibility to WSN and IoT [15]. They also point that the SDN approach applied to WSNs seeks to alleviate challenges concerning network management, and foster efficiency and sustainability in WSNs. Authors also argue that SDN would improve resource reuse, enable node retasking, establish and improve node and network management, as well as enable experiments with new protocols and to ease transition to standard protocols for deployed networks [6]. In particular, SDN is instrumental in dealing with device heterogeneity and infrastructure reuse in IoT deployments.

While the benefits SDN could bring to WSN and IoT are considerable, they will only be worthwhile once we have SDN southbound protocols that perform similar to standard routing protocols in WSN, such as the IETF RPL routing protocol (RFC 6550) [21]. Software Defined Wireless Sensor Networks (SDWSN) approaches are presented in the literature [15]. While several papers discuss the SDN approach for WSN and IoT, only a few proposals on SDN southbound protocols designed for WSN are found in the literature. Among these proposals, we highlight TinySDN [7] and SDN-Wise [10]. Given the limitations of the approaches available on the literature, we proposed IT-SDN [1, 19]. Unfortunately these papers present limited performance results, depicting small networks and limited data rates.

In this paper, we review IT-SDN [1] main features and present a performance evaluation aiming to fulfill this gap, comparing IT-SDN with RPL considering several network sizes and increased data rates (i.e., all the sensing nodes transmit data periodically). We conducted a number of experiments assessing the impact of flow table maximum capacity, and the metrics of data delivery, data delay, control overhead and energy consumption in order to show the advantages and trade-offs of using IT-SDN in comparison to RPL.

Our results show that IT-SDN is suitable for small network sizes, which could benefit from the flexibility pointed in the literature. The controller queue size is also varied in our experiments, and results hint this is another aspect that could benefit from improvements. We discuss the main challenges still faced by larger networks, and how they could be addressed to enable such benefits. We argue that our results and the directions pointed are an incentive to foster further SDWSN research, evaluating if improvements would increase IT-SDN performance overcoming its scalability issues.

This paper is organized as follows. Section 2 reviews SDWSN related work. A brief review of IT-SDN, our framework, is presented in Section 3. The experimental method is explained in Section 4, where we also discuss the metrics, as well as review RPL main features. Results and discussion are addressed in Section 5. Lastly, Section 6 presents final considerations and future work.

2 Related Work

As already discussed in the literature [19, 15], applying the SDN paradigm to WSN imposes different challenges and requirements when compared to wired networks. The main challenge is the limited resources available on the devices (i.e., energy, processing, memory, bandwidth and payload size). The requirements are related to the applications characteristics (e.g. sensing rate, data size, data transmission pattern and aggregation), as well to the nodes behavior due to duty-cycles, operating systems and programming approach.

Kobo *et al.* [15] present a comprehensive review of the SDWSN literature, discussing some of the challenges facing this paradigm, as well as the major SDWSN design requirements that need to be considered to address these challenges. Among the SDWSN approaches found in the literature, the ones based on OpenFlow [17, 18] have issues concerning the overhead introduced and the use of TCP as underlying communication protocol. IEEE 802.15.4 frame size is 127 bytes including header, what fragments TCP segments. Also, TCP suffers with wireless link failures and usually has issues trying to adjust the round trip time estimation. Furthermore, they were not implemented in WSN devices.

TinySDN [7] is a flow-ID-based approach that improves on previous work by enabling the use of multiple SDN controllers and by addressing (i) in-band control (i.e., control and data packets share the same bandwidth); (ii) higher communication latency; (iii) smaller link layer frames; and (iv) limited energy supply. TinySDN provides the code and related documentation, but it is highly dependent on TinyOS [13], thus limiting the platforms it could be deployed. Furthermore the use of the ActiveMessage function limits interoperability with other systems.

SDN-WISE [10] defines mechanisms for a stateful flow table, pursuing two goals: (i) to reduce the amount of information exchanged between sensor nodes and the SDN network controller, and (ii) to make sensor nodes programmable as finite state machines. SDN-WISE is also employed in a study of SDWSN in the presence of mobile nodes and multicast packet transmissions [3]. Nonetheless, the traffic conditions are unrealistic, as at most two nodes transmit data packets. Lasso *et al.* [16] recently proposed a software-defined networking framework for IoT based on 6LoWPAN. Unfortunately the paper lack details on the architecture and implementation, but they present energy consumption results for their implementation running on Contiki OS [9].

Given the limitations of the approaches available on the literature, we proposed IT-SDN [1]. IT-SDN is an SDWSN tool that is completely open and available. While its design is inspired by TinySDN [7], we improved the architecture, protocols and implementation.

3 IT-SDN Framework

When designing IT-SDN [1] we considered that the SDWSN architecture contains three main communication protocols: southbound (SB), neighbor discovery (ND) and controller discovery (CD). The SB protocol is used for the communication between controller and SDN-enabled devices. It defines the packet format, how these packets are processed and the state machine. The ND protocol is used to obtain and maintain node neighborhood information. Lastly, the CD protocol identifies a next hop candidate to reach the controller.

IT-SDN underlying architecture is independent of the operating system and its functions, what is key to enable interoperability. IT-SDN also defines a clear separation of the protocols used to achieve SB, ND, and CD, what creates the environment to evaluate new protocols to achieve these tasks depending on the network characteristics. The message exchange between IT-SDN node and the controller includes: neighbor report messages and flow related messages. Neighbor report messages enable the controller to create the centralized network view, which is employed to calculate flow rules.

The flow rules are installed in the nodes flow tables, the maximum capacity of which is an important parameter concerning the protocol scalability, as further discussed in this paper. When a node receives a packet for which it does not have a matching rule in its flow table, it sends a flow request to the controller. Next, the controller calculates the action the node should execute according to the current policy and enqueues the corresponding flow setup message in the transmission queue. After receiving the flow setup message, the node is able to process matching packets.

4 Experimental Method

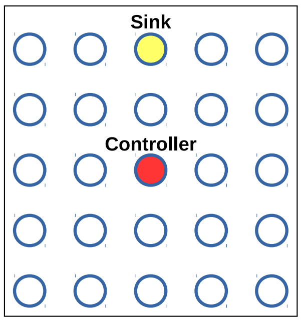
In order to evaluate IT-SDN[1] performance, we conducted a number of experiments varying the number of nodes. To carry out the simulations we used COOJA [20], a WSN simulation tool which is able to emulate the hardware (specifically msp430-based platforms) and to simulate the radio medium. The MAC layer is the IEEE 802.15.4 standard [14] using the ContikiMAC duty-cycle mechanism with the channel check rate set to $16Hz$ [8]. We chose ContikiMAC as it is Contiki's default duty-cycle mechanism, and it is easy to integrate with upper layers due to its asynchronous, statelessness and local operation characteristics. We simulated six square grid topologies, varying from 16 to 81 nodes. We chose grid topologies since they are easy to reproduce, and provide many possible paths. While some real deployments may not follow such pattern, deployments in cities or buildings usually follow a similar structure.

The topologies were configured with one sink and one controller. The sink and the controller position was set according to the following rules: (1) the controller and the sink are always in the same column; (2) for 25, 49, and 81 node-topology, the controller is at the center of the grid and the sink is in the top row (as depicted in Figure 1a); (3) for 16, 36, and 64 node-topology, the controller and the sink are to the left of the center of the grid (as depicted in Figure 1b). The controller is placed at the center to lessen the average number of hops to the other nodes. We chose to separate the sink from the controller since they are not necessarily the same entity, as considered by some SDWSN architectures.

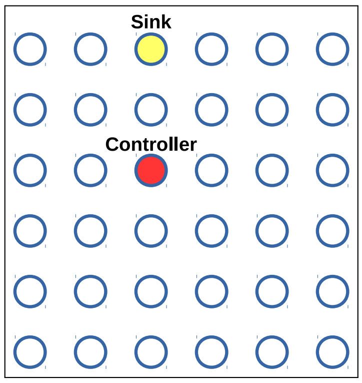
All nodes in the network transmit constant bitrate (CBR) data at 1 *packet/min*, excluding the sink and the controller. The data packets consist of the MAC and routing layers headers plus a 10-byte application payload. CBR traffic is typical for modeling periodic monitoring applications, while 10 bytes is enough to store simple measurement data. Collecting at 1 *packet/min* is deemed to provide sufficient data granularity without straining the sink node, although the specific requirements vary according to the application.

Another two important parameters are the flow tables and the controller's packet buffer size. All the topologies were simulated using flow tables of 15 and 30 entries, and a controller packet buffer of 100 packets. The 81-node topology was also simulated increasing the packet

[1] We used IT-SDN version 0.3 for our evaluation, available at `http://www.larc.usp.br/users/cbmargi/www/it-sdn/`



(a) 25 nodes topology



(b) 36 nodes topology

Figure 1: Sink and controller position examples

buffer to 255 packets.

Since IT-SDN is implemented on Contiki OS, we used the Collect Protocol [12] available in this operating system as the Neighbor Discovery protocol, thus ETX [4] is used as link metric. Using ETX provides a fair comparison, as it is the same metric used by RPL objective function. The neighbor information is transmitted to the controller using neighbor report packets. To limit the transmission rate of these packets, the SDN layer uses two rules: (i) a maximum of 1 neighbor report per minute, and (ii) the report is transmitted only if there is a significant change in the neighbor metric. A significant change means a relative difference of 200% or absolute difference of 100% from any previous link metric measurement.

The controller actively configures how the nodes will reach it. The configuration is based on the neighborhood information gathered. A simple reliability mechanism by periodic packet retransmission is implemented for control packets. The controller determines the shortest path using the Dijkstra algorithm. A new route is installed in the nodes if, after receiving a neighbor report, the new route cost is at least 20% smaller. This threshold avoids unnecessary flow setups due to temporary instabilities in the link quality, a mechanism similar to the hysteresis employed by RPL objective function.

The results of IT-SDN performance were compared with the IETF RPL routing protocol [21] performance running on similar scenarios. We chose RPL because it is an efficient standardized protocol, which is already implemented in Contiki OS. A brief explanation of RPL is provided in Section 4.2. Table 1 summarizes the parameters used in the simulations. Each topology scenario was simulated 10 times, each one running during one hour. The graphs in Section 5 show the results with a 95% confidence interval, representing the variability among the replications.

4.1 Performance Metrics

The IT-SDN performance analysis is based in four metrics: delivery rate, delay, control overhead, and energy consumption. The total delivery rate metric considers only the data packets. This metric is calculated dividing the total number of packets successfully received (by the sink) by the total packets sent by the application layer of all the nodes in the network.

The total delay is an average of the time the data packets spent to reach the destination. This metric is calculated adding up all the individual delays and then dividing it by the total number of packets received. The metric calculation excludes the undelivered packets.

The control overhead represents the extra workload induced by the routing protocol in the network. For our experiments, this metric is quantified as the total amount of control packets transmitted. The energy consumption metric is defined as the average energy consumption of all the nodes, excluding the controller. The energy consumption is calculated using a four-state model. This model considers the processing, transmitting, receiving and sleep mode energy consumption.

Thus, we use Equation 1 to calculate the energy consumption of each node, where P_i is the average energy consumption of the state i and T_i is the time spent on state i. Each of the P_i values are calculated according to the energy consumption parameters presented in Table 1. The voltage and current consumption values were obtained from the CC2420 datasheet, the radio used by the TelosB motes. Then, using Equation 2 we calculate the average energy consumption of the network, where E_j is the energy consumption of the node j and n is the number of nodes considered.

$$E_{node} = \sum_{i=1}^{4} P_i T_i \tag{1}$$

$$E_{avg} = \frac{\sum_{j=1}^{n} E_j}{n} \tag{2}$$

Table 1: Simulation parameters

Simulation parameters	
Topology	Square grid
Number of nodes	16, 25, 36, 49, 64, 81
Node boot interval	$[0, 1]$ s
Number of sinks	1
Data traffic rate	1 packet per minute (CBR)
Data payload size	10 bytes
Data traffic start time	$[2, 3]$ min
ContikiMAC channel check rate	16 Hz

Energy consumption parameters	
Transmission current consumption	21,7 mA
Receiving current consumption	22 mA
Processing current consumption	2,33 mA
Sleeping mode current consumption	0,180 mA
Operation voltage	3 V

IT-SDN parameters	
Controller retransmission timeout	60 s
ND protocol	Collect-based
Link metric	ETX
Neighbor report max frequency	1 packet per minute
CD protocol	none
Route calculation algorithm	Dijkstra
Route recalculation threshold	20%
Flow table size	15 or 30 entries

RPL parameters	
MOP	storing
RPL instances	1
Minimum DIO interval	4.096 s (contiki default)
DIO interval doubling	8 (contiki default)
DAO latency	4 s (contiki default)
DAO expiration	60 s (contiki default)
Objective Function	Minimum rank with hysteresis
DODAG metric	ETX

4.2 RPL Protocol

RPL is the IPv6 routing protocol for Low Power and Lossy Networks (LLN) defined in IETF RFC6550 [21]. It was designed on top of 6LoWPAN to provide efficient routing for the Internet of Things and it supports three traffic flows: point-to-point, point-to-multipoint, and multipoint-to-point.

RPL is a distance vector routing protocol, which is based on the concept of Directed Acyclic Graphs (DAGs). To construct the routes, RPL represents the topology as a DAG. Then, this DAG is partitioned into one or multiple Destination Oriented DAGs (DODAGs), where each DODAG is routed toward one root (which is expected to be a data sink) [21].

All nodes within the DODAG have a rank. The rank defines the nodes position relative to other nodes and to the DODAG root. The rank of the DODAG root is set to MinHopRankIncrease. The other nodes select their parent and then calculate their own rank according to the parent's rank plus the objective function. The objective function used is the Minimum Rank with Hysteresis, defined in IETF RFC6719 [11]. Then, the nodes inform the rank to their neighbors using DIO (DODAG Information Object) control messages. When all the nodes have selected their parent, the DODAG construction is concluded and the nodes are able to reach the root.

5 IT-SDN Performance Evaluation

In this section we present the results from our performance evaluation, as well as its analysis and discussion. The first set of results presented focus on the RPL and IT-SDN comparison for 16 – 81 network sizes, where two sets of configurations were considered for IT-SDN: flow tables with 15 entries and controller buffer with 100 entries, and flow tables with 30 entries and controller buffer with 100 entries. Second, we analyze the impact of the controller buffer on the performance considering the 81-node network and simulated the buffer size with 100 and 255 entries. The third aspect of this analysis focus on IT-SDN tradeoffs when increasing network size and the parameters (timers, flow table size, buffers) with most impact.

5.1 RPL and IT-SDN Comparison

The first metric we present is data delivery rate. As depicted in Figure 2, for smaller network sizes RPL and IT-SDN delivery rate is very similar. RPL achieves 99.9% on all scenarios, while IT-SDN shows more variations. Notice that the flow table size directly influences IT-SDN scalability, since data delivery rate is higher than 94% for up to the 64-node scenario when the flow table entry size is 30.

The nodes neighboring the controller need at least $\frac{n}{4}$ flow table entries. If there is not enough entries, the nodes farthest from the controller will not join the network, as the implemented policy is to ignore flow addition attempts whenever the flow table is full. Therefore, some nodes will not be able to deliver data packets, decreasing the overall delivery rate. The flow table management policy should be evaluated in order to improve scalability.

The second metric evaluated is delay, shown in Figure 3. Once again, IT-SDN is competitive to RPL at small networks (up to 36 nodes). A factor that increases IT-SDN overall delay is the time necessary to deliver the first packet. In order to depict such impact, we added two curves to the graph showing the data delay for packets transmitted after 20 minutes from the beginning of the simulation execution. This impact is more pronounced for larger networks, since the convergence time is larger. Also, we did not artificially increase the application traffic start time as we consider that the application layer is independent from the routing layer, enabling a more realistic analysis.

For IT-SDN, standard deviation increases with network size. This occurs as single events, such as failing to delivery a flow setup message, may greatly increase the first packet(s) delivery interval. After the initial delay, IT-SDN delivers nearly as timely as RPL.

Next we evaluated the energy consumption. As depicted in Figure 4, IT-SDN spends about 12% more energy than RPL. Even in scenarios where IT-SDN and RPL have the same amount of control packets, IT-SDN spends more energy. This occurs due to the fact that IT-SDN sends more broadcast packets than RPL, thus consuming more energy due to ContikiMAC behavior concerning broadcast packets. For the 15-entry flow table size, there are nodes that do not join the network. Therefore the energy spent decreases since less packets are transmitted (only neighbor discovery packets).

Concerning control overhead, the number of packets is shown in Figure 5. For small networks (up to 36), IT-SDN has a similar number of control packets as RPL. It is worth noting that IT-SDN needs end to end control communication between each node and the controller. In larger topologies, the path gets longer thus the chances of packet drop also increase. Whenever a packet that requires reliability is lost (e.g flow setup), it needs to be retransmitted, increasing the overall control overhead.

5.2 Impact of the Controller Buffer

As described in Section 3, when the controller receives a flow request it calculates the action the node should

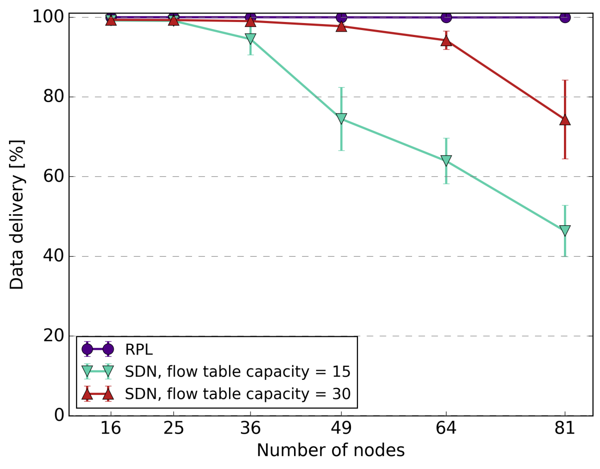


Figure 2: Data delivery results

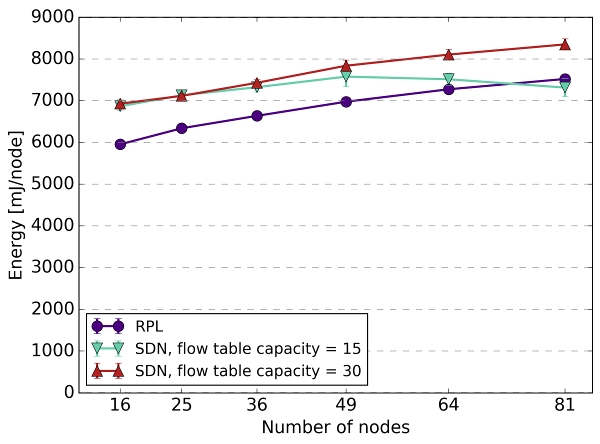


Figure 4: Energy consumption results

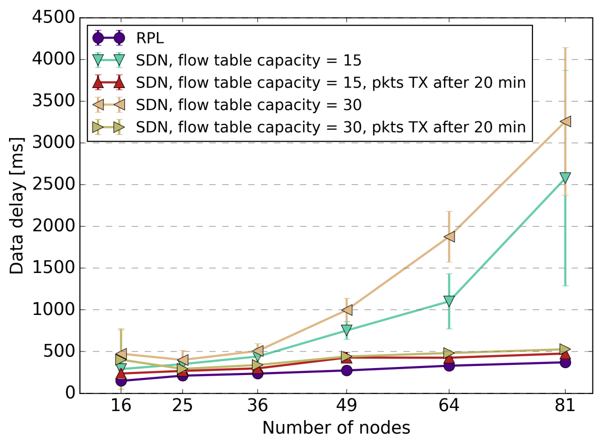


Figure 3: Data delay results

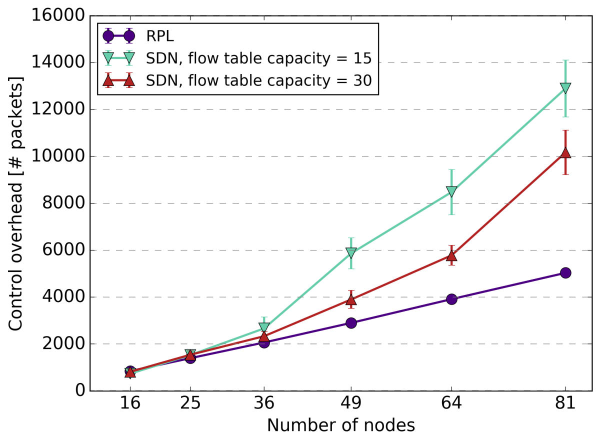


Figure 5: Control overhead results

execute according to the current policy, and then enqueues the corresponding flow setup message in the transmission queue. We would like to understand if this queue size would impact IT-SDN performance.

Table 2 depicts the delivery rate as well as average delay for the 81-node network simulation with buffer size of 100 and 255 entries. We expected that increasing the controller queue size, less packets would be discarded and thus we would achieve a better delivery rate and smaller delay. Considering only the successful simulation runs, *i.e.* the ones with data yield, minor gains were achieved. However, 2 out of the 10 runs for the 255-entry queue size simulations delivered no data at all, as the sink was not able to join the network. Therefore the average delivery did not improve considering all simulation runs.

These results hint that other characteristics of the controller queue could be explored. For instance, we currently employ a first-in-first-out enqueuing policy, while a higher priority could be assigned to packets destined to nodes near the flow destination. Another factor that increases the controller queue length is the periodic retransmission mechanism for reliability. More sophisticated schemes could be designed considering dynamic retransmission interval, considering the node distance from the controller, network congestion or likelihood of node mobility.

Table 2: Effect of controller queue size

Controller queue size	Delivery Rate	Delay [ms]
100	74.3%	3258
255 (all runs)	61.4%	3111
255 (successful runs)	76.7%	3111

5.3 IT-SDN tradeoffs

In order to achieve the benefits SDN could bring to WSN and IoT (improved management, increased flexibility and sustainability), we need to have SDN southbound protocols that perform similar to standard routing protocols in WSN, such as the IETF RPL routing protocol (RFC 6550) [21].

These performance evaluation results show that IT-SDN is suitable for small network sizes, delivering results similar to RPL for data delivery rate, delay, energy consumption and control overhead. We focused on the performance evaluation of convergecast traffic, a pattern which RPL was optimized for, and that is typical of WSN deployments.

When network sizes increase, the challenges SDN faces with scalability appear. All the evaluated metrics show a decrease in terms of performance. The main factors that contribute to these challenges are: reliable end-to-end control channels, flow table sizes, and the time it takes for first packet delivery. To address such challenges, we would need specific mechanisms. Concerning flow table sizes, we could use flow table compression and flow table entry replacement criteria to improve its usage and decrease the overhead introduced by entry replacement.

To decrease the initial delay for sending data, proactive flow setup would have a significant impact, but it depends on the applications to initiate the flow configuration. Configuring multiple nodes with a single packet, a feature that was envisioned in previous work but not implemented [7, 10], would also decrease the initial delay. To address the need of a reliable end-to-end control channel, we could follow the approach proposed by Baddeley *et al.* [2], isolating IT-SDN control traffic with layer-2 slicing in 6TiSCH or a similar approach using directly the TSCH mode from IEEE 802.15.4.

In summary, IT-SDN is suitable for small network sizes, which could benefit from the flexibility pointed in the literature. Concerning larger network sizes, one approach that could be taken to achieve better performance is to organize the large network in smaller clusters (for example with 25 nodes each) and use distributed and hierarchical controllers, as proposed by Spotled architecture [5]. Since performance evaluation presented similar results for both RPL and IT-SDN for small network sizes, one could ask why not use RPL then. We argue that the SDN benefits (improved management, increased flexibility and sustainability) are the main reasons to choose IT-SDN over RPL.

Improved management concerns not only network resources (i.e., bandwidth and buffer allocation), but also node management (e.g., battery level, tasks being executed) and application management (the application manager knows which tasks are being executed by each node, what data is sent to which sink, what is the sensing rate, what sensors are used by each task). The centralized view of the SDN controller could be used by a service orchestrator to make decisions about new tasks admission. For instance, in the shared infrastructure scenario [19], once there is a request for a new application, the service orchestrator would follow these steps: (i) verify with the application manager if the nodes could run the incoming task and provide the desired data rate, considering their energy level and already existing tasks; (ii) verify with the SDN controller which are the routes to the application sink; (iii) verify if the new routes to the sink will affect the current network performance. Depending on the previous steps, the new application could be admitted. This example depicts not only the improved management but also the flexibility that SDN aims to bring.

Another aspect that could be explored by the SDN controller is to use information concerning the underlying communication technology to select the path the package will traverse. For instance, in IoT heterogeneous scenarios, devices could have two radios (e.g. IEEE 802.15.4. and BluetoothLE). The SDN controller would know about both radios and use the different link layers to select the routes that match the application requirements.

Therefore we argue that results in this paper and SDN benefits are good enough to foster further SDWSN research, evaluating if improvements we discussed would increase IT-SDN performance, overcoming scalability issues depicted in the performance evaluation. Once this performance gap is covered, the service orchestrator architecture becomes a key issue to be addressed.

6 Conclusion

Several authors have highlighted how SDN would bring flexibility to WSN and IoT [15]. While these benefits are considerable, they will only be achieved once SDN southbound protocols perform similar to standard routing protocols in WSN.

To address such gap, we use IT-SDN [1] to conduct a performance evaluation comparing it with RPL. We considered several network sizes and increased data rates (i.e., all the sensing nodes transmit data periodically), conducting a number of experiments varying the number of nodes and assessing the impact of flow table maximum capacity. We assessed the metrics of data delivery, data delay, control overhead and energy consumption in order to show the advantages and trade-offs of using IT-SDN in comparison to RPL.

Results show that IT-SDN is suitable for small network sizes. For larger networks, we argue that our results are an incentive to foster further SDWSN research, evaluating if improvements would increase IT-

SDN performance overcoming its scalability issues.

Future work concerns designing and evaluating specific mechanisms to address the challenges related to scalability. The challenges are related to: reliable end-to-end control channels, flow table sizes, and the time it takes for first packet delivery. Also the controller queue should be explored, employing different enqueuing policies, and improving the periodic retransmission mechanism used for reliability.

ACKNOWLEDGMENT

Cíntia B. Margi is supported by CNPq fellowship (Pq) #307304/2015-9. Renan C. A. Alves is supported by FAPESP PhD fellowship number 2016/21088-1. Gustavo A. Nunez Segura is supported by the CAPES-PROEX fellowship #0212083 and the Universidad de Costa Rica.

REFERENCES

[1] R. C. A. Alves, D. A. G. Oliveira, G. A. Núñez, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," in *SBRC 2017 - Salão de Ferramentas*, May 2017. [Online]. Available: http://www.larc.usp.br/~cbmargi/it-sdn/it-sdn-sbrc2017sf.pdf

[2] M. Baddeley, R. Nejabati, G. Oikonomou, S. Gormus, M. Sooriyabandara, and D. Simeonidou, "Isolating sdn control traffic with layer-2 slicing in 6tisch industrial iot networks," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 247–251.

[3] C. Buratti, A. Stajkic, G. Gardasevic, S. Milardo, M. D. Abrignani, S. Mijovic, G. Morabito, and R. Verdone, "Testing protocols for the internet of things on the euwin platform," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 124–133, Feb 2016.

[4] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wirel. Netw.*, vol. 11, no. 4, pp. 419–434, Jul. 2005.

[5] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined wireless sensor networks," in *2016 IEEE International Symposium on Consumer Electronics (ISCE)*, Sept 2016, pp. 85–86.

[6] B. T. de Oliveira, R. C. A. Alves, and C. B. Margi, "Software-defined wireless sensor networks and internet of things standardization synergism," in *Standards for Communications and Networking (CSCN), 2015 IEEE Conference on*, October 2015, pp. 96–101.

[7] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*, Nov 2014, pp. 1–6.

[8] A. Dunkels, "The contikimac radio duty cycling protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, 2011.

[9] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 455–462.

[10] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE : Design , prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 513–521, 2015.

[11] O. Gnawali and P. Levis, "The minimum rank with hysteresis objective function," RFC 6719 (Proposed Standard), Internet Engineering Task Force, Sep. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6719.txt

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. New York, NY, USA: ACM, 2009.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, 2000.

[14] IEEE, "802.15.4-2011 - IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans)," 2011.

[15] H. I. Kobo, A. M. Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[16] F. F. J. Lasso, K. Clarke, and A. Nirmalathas, "A software-defined networking framework for iot based on 6lowpan," in *2018 Wireless Telecommunications Symposium (WTS)*, April 2018, pp. 1–7.

[17] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor openflow: Enabling software-defined wireless

sensor networks." *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[18] A. Mahmud and R. Rahmani, "Exploitation of openflow in wireless sensor networks," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, vol. 1, 2011, pp. 594–600.

[19] 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, special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2017) in conjunction with the VLDB 2017 Conference in Munich, Germany. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:101:1-2017080613467

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

[21] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks," RFC 6550 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

AUTHOR BIOGRAPHIES

Cíntia Borges Margi is Associate Professor at Universidade de São Paulo since 2015, where she started as Assistant Professor in February 2007. She obtained her degree in Electrical Engineering (1997) and her MSc in Electrical Engineering (2000) at Universidade de São Paulo, her PhD in Computer Engineering at University of California Santa Cruz (2006), and her Habilitation (Livre Docência) (2015) in Computer Networks from the Universidade de São Paulo. Her research interests include: wireless sensor networks (protocols, systems, security, energy consumption and management, embedded hardware) and software-defined networking.

Renan C. A. Alves is a PhD student at Universidade de São Paulo. He obtained his MSc degree (2014) and graduated in Electrical Engineering with emphasis in Computer and Digital Systems (2011) both from Universidade de São Paulo. His main research interests include network protocol modeling and performance analysis, Wireless Sensor Networks protocols and applications.

Gustavo A. Nunez Segura is a PhD student at Universidade de São Paulo. He received the MSc degree (2018) in Electrical Engineering from Universidade de São Paulo and the B.Sc in Electrical Engineering from Universidad de Costa Rica. His main research interests include energy consumption and security in wireless sensor networks and software-defined networking.

Doriedson A. G. Oliveira is a MSc student at Universidade de São Paulo since 2015. He obtained his Technician degree (2015) in Digital Games at FATEC Carapicuíba. His main research interests include: wireless sensor networks (protocols and routing) and artificial intelligence (swarm intelligence).