Distributed Control Plane Architecture for Software-Defined Wireless Sensor Networks

Bruno Trevizan de Oliveira and Cíntia Borges Margi
Escola Politécnica – Universidade de São Paulo (USP)
São Paulo - SP, Brazil
Email: {btrevizan, cbmargi}@larc.usp.br

***Abstract*—Wireless Sensor Networks (WSN) are the basis for several applications including environmental and health monitoring, industrial and home automation, consumer electronics connectivity, and public safety, being a key technology for Internet of Things and Smart Cities development. Software-defined networking (SDN) paradigm could reduce complexity in WSN configuration, management and re-tasking. However the use of a centralized controller increases WSN traffic, decreasing its lifetime. The SDN controller could leverage from the WSN local knowledge to improve the decisions taken. We propose an architecture with distributed and hierarchical controllers, where local controllers use local information to reply to nodes in its area, and a global controller oversees the whole network. We present results from simulations based on emulated sensor nodes.**

I. Introduction

Software-defined networking (SDN) is a paradigm that uses a logically centralized software, hosted in nodes called SDN controllers, to control the behavior of a network, reducing the complexity of network configuration and management. The first initiatives to implement SDN have targeted wired networks. Specifically for wireless sensor networks (WSN), which we called software-defined wireless sensor networks (SDWSN), the main approaches include Flow-Sensor [1], Sensor OpenFlow [2], TinySDN [3] and SDN-WISE [4].

Typical WSN devices have only one radio that either transmit or receive in one given frequency at a given time, and thus in-band control communication between SDN nodes and SDN controllers is required. Scalability is a concern and one should limit the overhead of frequent events on the control plane. To achieve this, previous work considered the idea of control distribution in order to reduce the traffic control delay and processing overhead introduced by one centralized controller. Kandoo [5] proposes a hierarchy of controllers, where local ones have limited network view and deal with local applications. Devoflow [6] considers a logically centralized and physically distributed controller. Previous work on SDWSN consider only one centralized SDN controller.

We propose the distribution of SDN control software for WSN through the `Spotled` architecture, by using a WSN cluster-head as local controller in a given network neighborhood, which is physically close to the SDN-enabled nodes. In order to keep the overall network view, a global controller oversees the whole network, exchanging information with local controllers. To showcase our proposal, we present results from simulations for our SDWSN implementation.

II. Spotled Architecture

`Spotled` is an SDN controllers architecture designed to reduce control traffic. To achieve this goal two measures are adopted: physical distribution of controllers, and establishment of a 2-level hierarchy of controllers. It specifies three types of components: global SDN controller, local SDN controller and SDN-enabled sensor node.

The **SDN-enabled sensor node** is a combination of SDN-enabled switch and end-device. Its behavior is defined by flow tables and related actions. It receives messages from local controller that contains flows and actions specifications, and sends flow request when not able to handle a data packet. SDN-enabled sensor nodes are in the neighborhood of a local controller, either by one or multiple hops.

The **local SDN controller** acts as controller of a WSN subdomain. It is responsible for managing all SDN-enabled sensor nodes in its area. It sends actions specifications to the sensor nodes changing their flow tables, and notifies the global controller about relevant local topology changes.

The **global SDN controller** is a logically centralized controller that maintains the network-wide state, creates and manages flows between different local SDN controllers subdomains, both proactively or when required by local controllers.

Global SDN controller component does not interact directly with SDN-enabled sensor nodes, it send commands to the local SDN controllers, which in turn send specific actions to sensor nodes. Notice that a given node could act both as global and local controller.

III. Evaluation Methodology

We used TinySDN[1] as SDN-enabled sensor nodes and underlying SDWSN protocol implementation. `Spotled` SDN controllers were implemented as standalone software as a proof of concept and experimentation. The experiments were executed as simulations on COOJA[2], a cross-level WSN simulator that enables the use of the platform compiled binary code by emulating the TelosB[3] device.

The main goal is to measure and compare the time sender nodes take to established data flows and to send a message to the sink node, both in a SDWSN with a single controller

[1] http://www.larc.usp.br/~cbmargi/tinysdn
[2] https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja
[3] http://memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

978-1-5090-1549-8/16/$31.00 ©2016 IEEE

and using Spotled architecture with multiple controllers. In our test scenarios the controller nodes were also hosted on TelosB mote.

For the experiments we considered four different scenarios with seven *SDN-enabled sensor nodes*, where two of them generate the data packets to be sent to the sink node, four nodes perform only the forwarding task and, one node acts as a sink. The number of *SDN controller nodes* is variable according to the scenario proposed. Scenarios `A` and `B` depict a single controller SDWSN, and scenarios `C` and `D` illustrate Spotled architecture cases, i.e., a SDWSN with a distributed and hierarchical set of controllers.

Figure 1 illustrates the scenarios including the four cases, where nodes `1` and `7` are the senders. `Node 4` is the sink in scenarios `A`, `B` and `C`. In `scenario C` the sink node is reached by both local controllers subdomain, illustrating the case where local controllers do not need to communicate with the global controller to establish necessary data flows. For `scenario D` the sink is `node 5` and it is out of `node 0` local controller subdomain, and thus this local controller needs to communicate with the global controller to request a data flow beyond its subdomain.

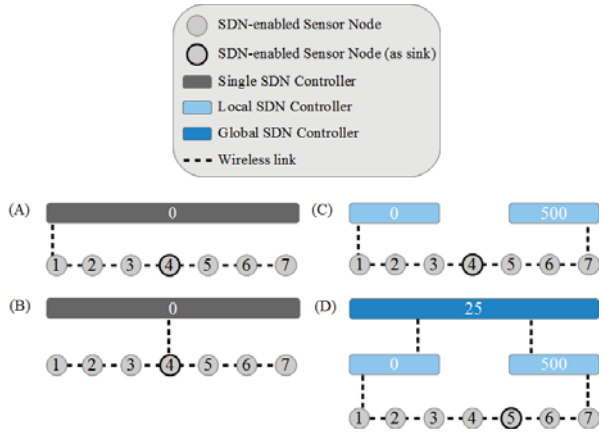

Fig. 1. Experimental Scenarios.

IV. Experimental Results

Figure 2 (a) depicts differences in time to controller assignment for sender nodes (considering the control flow specification to the controller in the SDN-enabled sensor nodes). Despite the scenario `B` presents a fairer balance than the scenario `A` due to the node controller position, average results for controller assignment in scenarios `A` and `B` are approximately the same (under 3.3 seconds). The average time to controller assignment considering both Spotled scenarios (`C` and `D`) is 1.2 seconds, i.e., less than 37% of the time when compared to single SDN controller scenarios, which occurs due to controllers distribution. Scenario `D` took a little longer than scenario `C` (1.33 seconds and 1.07 seconds, respectively), due to the greater number of nodes in the network, but it is less than 41% of the average time for scenarios `A` or `B`.

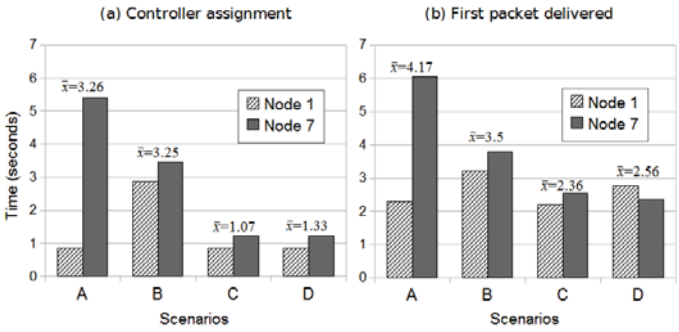

Fig. 2. (a) Time to controller assignment, and (b) Time to deliver first packet.

Figure 2 (B) presents timings for the first packet delivered from both sender nodes for all scenarios. The average times for both sender nodes to the sink, in comparison with the best results for a single SDN controller, Spotled architecture yielded a gain of approximately 33% for scenario `C` (in which local controllers does not need to communicate to global controller) and 27% for scenario `D` (in which the participation of the global controller is required).

V. Final Considerations

Experimental results show that the Spotled architecture is able to minimize the impact of control traffic introduced by the SDN paradigm in WSN, thus reducing the time to data flows installation on the network and the initial latency caused by the need of communication with a controller.

As future work, we plan to extend the experiments scenarios, increasing number of nodes in the network, including mobility and integration with different types of networks.

Acknowledgment

This work is funded by São Paulo Research Foundation (FAPESP) under grants #2013/15417-4 and #2014/06479-9. Cíntia Borges Margi is supported by CNPq research fellowship #307304/2015-9.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012.

[6] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "Devoflow: Cost-effective flow management for high performance enterprise networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX, 2010.