Software Defined Topology Control Strategies for the Internet of Things

Tryfon Theodorou and Lefteris Mamatas
Department of Applied Informatics,
University of Macedonia, Thessaloniki, Greece
Email: tryfonthe@uom.edu.gr, emamatas@uom.edu.gr

***Abstract*— Topology control is a crucial process for an efficient operation of a Wireless Sensor Network (WSN). The usage of WSNs in the Internet of Things (IoT) emerges new research challenges and novel applications. Recent WSNs proposals enhanced with Software Defined Networking (SDN) practices introduce new innovative network control strategies and protocols based on central control logic. This paper introduces two novel topology control techniques for Software Defined WSNs that can be combined and adapted to the context environment. In this direction, we propose CORAL-SDN, an SDN framework for WSNs that enables dynamic deployment and configuration of multivariate topology control mechanisms. We evaluate such topology control strategies using our own novel SDN experimentation facility for IoT. The results demonstrate significant improvements on WSNs management, control features and performance in terms of topology construction time and reduced topology maintenance overhead.**

Keywords*— *Topology control; Software Defined Wireless Sensor Networks

I. INTRODUCTION

WSNs consisting of tens or hundreds of wireless sensor nodes, called *motes*, are mainly used to measure and monitor real world phenomena of interest, in high precision and large scale. They are used in a wide range of applications such as tracking, surveillance, disaster detection, environmental monitoring, healthcare and agriculture. Their main characteristics are the limited computational processing power and memory, low power operation and resources, limited bandwidth, and low quality radio communication [1].

WSNs today are becoming a key-enabling technology for the Internet of Things (IoT) [2], introducing a new range of WSN applications integrated to the traditional Internet infrastructure. The transition of WSNs to the new era of IoT, introduces new challenges and imposes the exploration of novel ideas in terms of new applications. Major relevant issues are interoperability, heterogeneity, quality of service, and security.

An approach that targets the above challenges exploits new flexible network architectures, such as the Software-Defined Networking (SDN) which uses logically centralized software, hosted in nodes called SDN controllers, to control the behavior of a network by reducing the network configuration and management complexity. SDN was originally implemented for wired networks operating in cloud data centers. Recent research endeavors [3] are blending SDN and SDN-like architectures with WSNs technologies forming a new approach for SDNs called software-defined wireless sensor networks (SDWSNs). The SDWSN paradigm brings new ways in the WSNs control, management and operation. For the time being, the research community is mainly concentrating on SDN routing and data flow control. In this paper, we argue that a centralized SDN approach can improve WSNs aspects beyond routing and flow manipulation, for example the topology control.

Topology Control is one of the most important and critical procedures used in the operation of WSNs [4]. The goal of topology control is the formulation of a graph representing an abstract view of the network nodes, and their communication links. This abstract representation is mainly used in routing decisions aiming for fewer bottlenecks, reduction of traffic, low latency and efficient energy consumption.

Although topology control has received a lot of attention in the WSNs community where a number of solutions have been implemented [5], the major changes towards IoT and network softwarization that WSNs are going through, compels to further investigation. An example is the efficient integration of Internet protocols under the SDN paradigm, in regard to low energy and lossy WSNs environment. Here, we study topology control solutions that can be applied in the new software defined manifestation of WSNs. In our understanding, this is the only paper that targets in depth the study of topology control in SDWSNs.

Our investigation is conducted and evaluated using our own SDWSN framework, the CORAL-SDN, which enables dynamic deployment and configuration of multivariate topology control mechanisms. In our experiments, we used our SDN experimentation infrastructure which is a general facility for evaluating network control and protocol mechanisms for the IoT [6]. In this paper, we propose and evaluate experimentally two novel topology control techniques that can be combined and adjusted on-demand. Our main research goal is to improve WSNs management, control, and operation, through advanced topology construction algorithms that follow the SDN paradigm and achieve reduced time and control overhead.

The paper is organized as follows: Section II provides background information for topology control on WSNs, SDNs and SDWSNs. In section III, we present our novel topology control strategies for SDWSNs. Our testing environment for experimenting with alternative topology control techniques, namely the CORAL-SDN, is described in section IV. Section V presents our experimental evaluation results and Section VI summarizes the main outcomes and outlines future research directions.

II. Related Work

The topology control algorithms in traditional WSNs can be classified based on [4] as: (a) *homogeneous* algorithms that assume a simplistic approach where the nodes are using the same transmitting range; and (b) *non-homogeneous* algorithms where the network nodes are capable of different transmitting ranges. The latter approaches, depending on the information used in topology construction, can be subcategorized into the following distributed topology control protocols: (a) *location based,* like R&M and LMST [7], where node positions are known and can be used in a centralized manner; (b) *direction based,* like CBTC [8] and RHG [9], where nodes do not know their position, but they can estimate the relative direction of their neighbors; and (c) *neighbor based,* like KNeigh [10], and XTC [11], where nodes are aware only of the address IDs of their neighbors and some criteria like the link quality or distance.

RPL [12], the dominant distance vector protocol for WSNs, specifies the construction of a Destination Oriented Directed Acyclic Graph (DODAG), using an objective function and a set of metrics/constrains. RPL builds a logical routing topology graph as an abstraction of the actual network. The network administrator(s) can decide to activate multiple versions of it by using different criteria for each graph, e.g., power consumption or link quality.

Regarding the typical SDN proposals, topology discovery is a critical service provided at the controller layer. The de-facto OpenFlow Topology Discovery Protocol (OFDP) adapts the Link Layer Discovery Protocol (LLDP) with certain modifications, but performs topology discovery in fixed SDN environments mainly [13]. Recent research in OpenFlow topology control like [14], investigates OFDP and suggests new improved topology discovery approaches, that reduce significantly the amount of control messages for infrastructure SDNs. Moreover, the Open Networking Foundation Wireless and Mobile workgroup [15] currently determines architectural requirements and suggests extensions in the OpenFlow protocol for wireless and mobile domains, e.g., LTE and Wi-Fi. Here, we investigate SDN strategies for topology control that are not constrained by the OpenFlow protocol, but they can be integrated in its future evolution.

In the research front of the SDWSN approaches, authors in [16] suggest a theoretical OpenFlow based protocol for WSNs, addressing key technical challenges for Software-Defined WSN architectures. The application of SDN-inspired techniques in WSNs is described in [17], [18], while highlighting a number of flexibility and efficiency advantages but also challenges that should be addressed like the increase of control message overhead and energy efficiency. To overcome these challenges, recent proposals are using stateful routing tables and proactive routing decisions to reduce interactions with the controllers, and improve the flow-control decisions, such as the SDN-WISE [19]. In the above solutions, the topology control process is initiated by the controller in a procedure where the sink node advertises its existence to the neighbor nodes which, in turn, are initiating a flood of control messages to discover the rest of the nodes. In TinySDN [20] and Spotled [21], each node recognizing its neighboring nodes, transmits a set of topological information to the controller which constructs the topology.

All the previously referenced SDWNSs proposals are using topology control techniques similar to the ones used in traditional WSNs, i.e., there is a main focus on the routing algorithms and their associated overhead. Furthermore, the SDN approaches are currently focusing on infrastructure networks and only recently in mobile networks. In this paper, we support the idea of a thorough investigation in topology control bespoke to software defined WSN proposals in order to exploit the advantages of SDNs, while tackling the challenges of the WSN environment, e.g., limited resources and low quality of wireless medium. Indicatively, an SDN controller can adapt the topology discovery process to the context environment. In the next section, we discuss our approach to topology control, along these lines.

III. Topology Control in SDWSNs

Topology Control (TC) is divided into two core phases: *Topology Construction* and *Topology Maintenance*:

The *Topology Construction or Topology Discovery* phase is usually initiated when the network starts its operation. Its main goal is to construct an abstract representation of the network topology and nodes' connectivity. Depending on the TC protocol, the discovered information is stored either into each one of the network nodes, in case of a distributed protocol, or, into a central node, i.e., the sink, in case of a centralized approach. Although topology construction is unavoidably a time consuming process, there are occasions where its duration is crucial, e.g., an emergency disaster scenario.

The *Topology Maintenance* phase is the recurring process that maintains the integrity of network connectivity during network operation. The main task is to update the abstract network connectivity structure. A high rate of topology maintenance execution requests, increases the amount of network control messages and subsequently increases the network's Control Overhead (1).

$$Control\ Overhead = Control\ Packets\ /\ Total\ Packets \quad (1)$$

The network TC has a main objective: using low control overhead to provide the routing protocol algorithms with adequate information in order to achieve efficient message delivery, higher throughput, lower traffic, less bottlenecks and optimal use of energy.

In this paper, we introduce two new TC algorithms for SDWSNs. Their task is to feed the global controller with sufficient information to enable it to create the topology graph. This graph composes a global representation of the WSN and is used from the global controller to make efficient decisions on the data flow establishments. The global controller is capable of using each algorithm separately or in combination in order to achieve the best possible outcome.

We adapt a typical IoT network scenario consisting of a global controller operating on a desktop computer connected to the infrastructure network, and a set of static or mobile motes placed in various physical topologies.

In the following two subsections we describe in detail the two TC algorithms.

A. Topology control algorithm based on node advertisement

The first TC algorithm, *based on node advertisement*, is called TC-NA and described by the sequence diagram in Fig. 1. The controller initiates the topology discovery process by sending a topology discovery control packet to the first node.

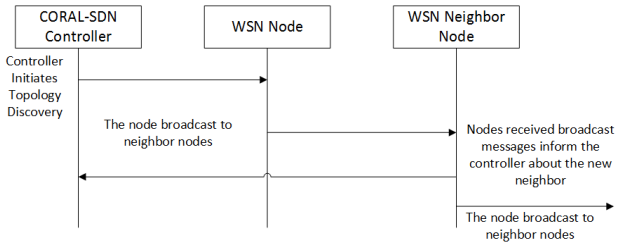

Fig. 1. The TC-NA algorithm

As described in Algorithm 1, when a node receives a control message from the controller, the latter is transmitting a broadcast beacon message advertising its location to the neighboring nodes. This control message includes various information related to the sender like the *node id*.

Algorithm 1 - TC based on node advertisement (TC-NA)

```
1: for all received packets pkt do
2:   if received pkt = advertisement broadcast from node then
3:     prepare replay.pkt add nodeid, received.pkt.nodeId,
                 link quality, signal strength, battery energy
4:       send replay.pkt to the controller
5:   end if
6:   if received pkt =  advertisement pkt from controller then
7:         send broadcast pkt to your neighbors
8:   end if
9: end for
```

The receiving node estimates data related to the signal strength, the link quality for the received message, and formulates a message with this information, adding also information related to its identification and its operational status, e.g., its battery power. This message is then transmitted back to the controller. When the controller receives a topology response message, it updates the network topology graph. Each node participates in the process, so consequently, the network floods from end to end.

B. *Topology control algorithm based on neighbor request*

We carry on with the second topology control algorithm that is *based on neighbor request*. The algorithm is called TC-NR and it is described by the sequence diagram in Fig. 2. The controller initiates the topology discovery process by sending a control message to the first node attached.

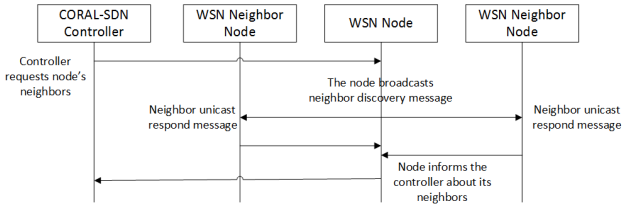

Fig. 2. The TC-NR algorithm

As described in Algorithm 2, when a node receives a control message from the controller, it is broadcasting a beacon message to all neighbor nodes in range. Each node that receives that broadcast message, responds to the sender with a unicast packet providing information regarding the communication link quality and address id. The receiver collects the respond messages from the neighbor nodes and informs the controller. The controller subsequently updates the network topology graph.

Algorithm 2 - TC based on neighbor node request (TC-NR)

```
1: for all received packets pkt do
2:   if received pkt = broadcast neighbor request then
3:       prepare replay.pkt add nodeid, link quality
                                  and signal strength
4:       send unicast replay.pkt to the sender node
5:   end if
6:   if received pkt =  pkt from controller then
7:       send broadcast pkt to your neighbors
8:   end if
9:   if node receives unicast pkt replay then
10:       prepare replay.pkt add nodeid, received.pkt.nodeId,
                link quality, signal strength, battery energy
11:      send replay.pkt to the controller
12:  end if
13: end for
```

C. *Comparison and discussion*

Prior to the quantitative evaluation in section V, we discuss the main architectural benefits and drawbacks of the above two algorithms. Initially, we have to acknowledge that both algorithms succeed to implement the topology discovery process. The TC-NA algorithm succeeds to collect the network information in a passive mode by reporting to the controller other nodes whenever these nodes advertise their existence. The TC-NR succeeds to collect the network information in an active mode, as each node requests an answer from its neighbors.

Comparing the amount of tasks each algorithm executes in relation to packets sent, the TC-NR algorithm shows a higher number of executed tasks. Consequently, we argue that the TC-NR algorithm produces inferior time performance results compared to the TC-NA, especially during the topology construction phase.

Considering the topology maintenance phase, we can point out that the TC-NA algorithm has to be executed exhaustively for all nodes in order to collect all possible neighbors. On the contrary, the TC-NR can acquire the neighbors of a node by directly requesting the node itself. This difference can be valuable in certain cases: e.g., in a heterogeneous network with mobile and static nodes, the TC-NR can execute topology requests on specific nodes or parts of the network, as many times as needed, without overloading the rest of the network with unnecessary topology control packets.

D. *Proposed improvements*

Based on the above discussion, we can easily conclude that depending on the network scenario (e.g., the network topology or application), the two algorithms have such strong and weak points. We suggest that a WSN should support both the TC-NA and TC-NR algorithms, and dynamically apply and configure each one on the right occasion. Using the software defined control paradigm over a WSN, provides a solution to the problem of coexisting algorithms. SDWSN centralized intelligence can handle complicated decisions utilizing the global network view and applying different solutions depending on the network context.

In the next section, we present our integrated SDWSN framework that is allowing us to experiment with alternative protocol strategies in various network scenarios.

IV. THE CORAL-SDN ARCHITECTURE

To experiment with the above algorithms, we implemented an innovative testing framework named CORAL-SDN. The CORAL-SDN enables elastic network operation through offloading complexity from the network protocols to the controller plane deployed at the surrounding fixed infrastructure. It supports deployment and dynamic configuration of alternative end-to-end topology control algorithms for IoT devices operating in heterogeneous environments, using centralized network control that exploits the global picture of the network. The facility is capable of taking into account fundamental characteristics of wireless networks, such as signal issues and intermittent connectivity. The CORAL-SDN establishes an ideal testing environment facilitating experimentation with software defined strategies and solutions for WSNs.

The CORAL-SDN framework, as part of the second WiSHFUL (http://wishful-project.eu) Competitive Call for Experiments, has been integrated with the WiSHFUL platform (https://github.com/wishfulproject) [22] and is capable of conducting experiments using the IMEC w-iLab.2 test-bed or the Cooja emulator. CORAL-SDN architecturally is organized in two subsystems: the CORAL controller and the WiSHFUL project infrastructure. Fig. 3 depicts the CORAL architecture and the interfaces between the different components.

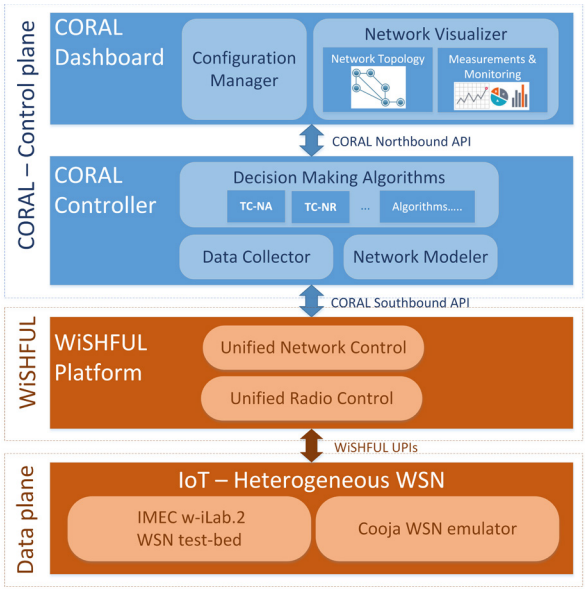

Fig. 3. CORAL-SDN Architecture

The *CORAL Controller* module, acting likewise an SDN controller, is responsible for the centralized management of the network flow control. The controller's intelligence comes from the *Decision Making* subsystem that specifies a set of rules and thresholds, organized in algorithms (e.g. the TC-NA and TC-NR algorithms). These algorithms implement different network functions like topology control and routing. Based on modular architecture, it easily adapts new algorithms that facilitate new functionalities. Another essential module of the controller, the *Network Modeler*, maintains an abstract view of the network connectivity in a graph structure incorporating a variety of cross layer data, including the signal strength and the link quality estimation measurements collected from the network.

On the northbound, the CORAL controller is connected with the *Dashboard* (Fig. 4), which constitutes a highly flexible GUI visualization tool based on the NODE-RED platform (https://nodered.org). The interface visualizes the WSN topology and presents various measurements provided by the controller, while it also offers management functionalities and configuration parameters to the user, such as the type of the applied topology control algorithm.

On the southbound, the controller communicates with the data plane through the WiSHFUL platform using the Universal Programming Interfaces (UPIs), i.e., novel lower protocol stack abstractions introduced from the WiSHFUL project. The control messages formed in JSON objects received from the WiSHFUL platform are propagated to the network nodes through a separate control channel. Using the WiSHFUL platform provides to the CORAL-SDN architecture heterogeneity on the data plane and robust experimentation facilities fully aligned with the software defined approach we are investigating.

CORAL-SDN is the extension of our framework presented in [6] aiming the dynamic configuration of routing algorithms through an SDN-inspired facility for the IoT.

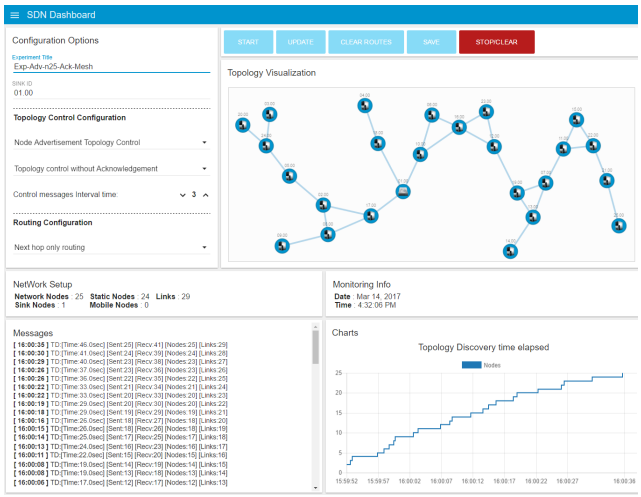

Fig. 4. The CORAL-SDN Graphical User Interface (Dashboard)

V. EVALUATION

The main goal of this section is to highlight our experimental results which compare the success rate and time needed for the controller to construct the network topology in different physical topology scenarios. Our evaluation focus on the topology discovery process rather than on the topology maintenance, because the topology control aspect is complex enough to deserve independent study. The latter constitutes part of our next research goals, integrating mobility issues as well. Thus, we firstly elaborate on our experimental setup, and then we present and discuss our results.

A. Experimental Setup

We implemented the SDWSN control plane of CORAL-SDN and our topology control algorithms (i.e., imported as controller modules) in Java. In the data plane, we implemented a multi-hop forwarding network protocol for Contiki OS (www.contiki-os.org) using the C programming language.

For our experimental evaluation, we used Cooja simulator with emulated Zolertia Z1 IoT devices. Cooja is a Linux based

cross-layer WSN simulator for Contiki OS, which enables the creation of virtual WSN scenarios. The radio environment was set to operate on channel 26 with channel check rate on 128 Hz. The control message packet size was 60 bytes.

We conducted our experiments with WSNs consisting of 25 nodes. We considered six distinct typical physical topology scenarios: Linear, Ring, Grid, Tree, Mesh dense, and Mesh sparse. Each scenario reflects different behavior regarding the network topology discovery. All experiments were conducted 10 times and the results demonstrated correspond to the mean value. This number deemed appropriate for the statistical accuracy of our analysis, i.e., produced a low standard deviation of our measurements.

The practical experience gained from the experimentation with our framework revealed a number of technical issues that have to be considered in SDWSN topology discovery strategies and are discussed in the paper. One of these issues was the detection of a certain amount of collisions, when the algorithms were initially flooding the network with topology control messages. To overcome this problem, we introduced a dynamically configured parameter in our algorithms specifying the time nodes have to wait before transmitting control messages. In order to evaluate the effect of this time interval, we conducted our experiments using two different ranges of randomly assigned values. The first one is in-between 1 to 3 sec, and the latter is in-between 1 to 6 sec.

B. Experimental Results

In Fig. 5, we present the topology construction duration in seconds, for each one of the experiments. We observe that linear topology produces the highest values as expected, since it investigates nodes one by one. On the contrary, the tree topology demands the minimum time. The results show clearly that the TD-NA algorithm is faster compared to the TD-NR. This outcome justifies our comments in section III-C. Regarding the time interval we conclude that an interval from 1 to 3 sec produces better results than the one from 1 to 6 sec. Some irregularities can be observed with the mesh high-density algorithm, as this topology is demanding in terms of the number of detected links, because of increased radio collisions.

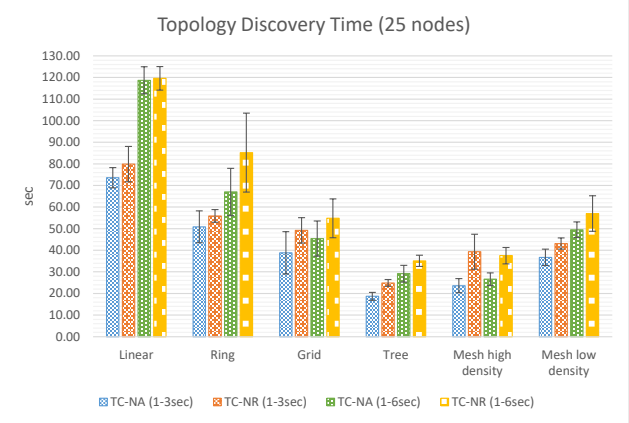

Fig. 5. Topology discovery duration for 25 nodes

In Fig. 6 to Fig. 9, we investigate the quality of the produced result compared to the algorithm's success rate. Fig. 6 shows that the node discovery, in general, is successful since the success rate is above 90% in all cases.

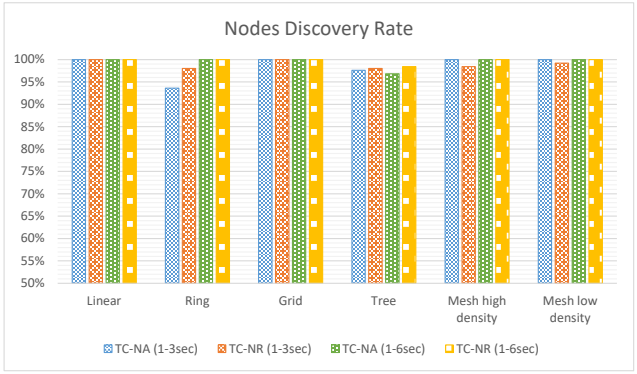

Fig. 6. Node discovery rate for 25 nodes

Fig. 7 depicts in detail the number of discovered nodes in each case. The tree topology and in some cases the ring topology scenario experienced some minor difficulties in recognizing the network. This is mainly due to the big amount of control messages exchanged in short interval.

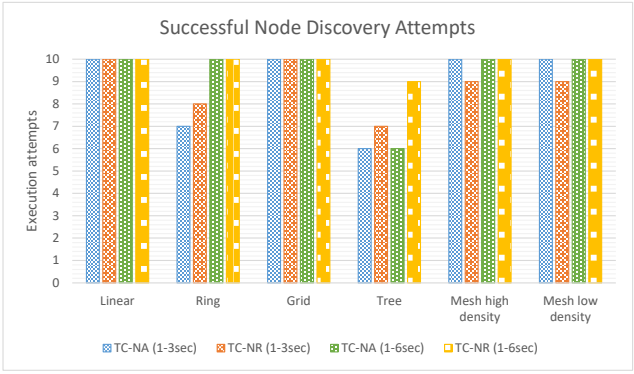

Fig. 7. Node discovery successful attempts for 25 nodes

In Fig. 8, the percentage of the successfully recognized connectivity links between the nodes is presented. From this chart we can conclude that link detection is even more challenging, leaving space for improvement for both algorithms. Although in most of the scenarios the unsuccessful link discovery ratio was less than 10%.

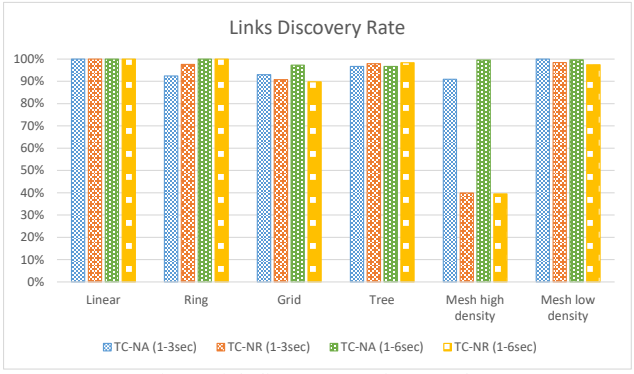

Fig. 8. Link discovery rate for 25 nodes

Fig. 9 shows the number of the experimental runs where successful detection for all network links is occurred. In some topologies, like the grid and the mesh dense, we observed the most challenging results, leaving space for improvement and further investigation. We note that well-established WSN algorithms (e.g., RPL) face similar issues as well. Our choice of topologies stress-test the topology discovery.

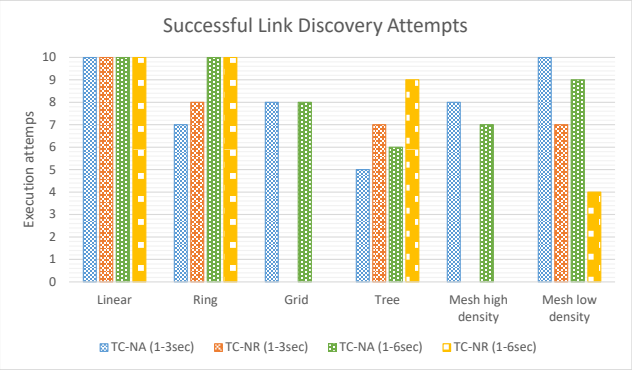

Fig. 9. Link discovery successful attempts for 25 nodes

Based on the evaluation outcome discussed above, the proposed SDWSN topology control algorithms exhibited a successful operation regarding the node discovery process and performance, i.e., rates above 90% in the majority of the topology cases. Moreover, the communication links discovery was almost 100% successful in most cases. Only in two cases the TD-NR was no able to detect all the communication links due to the amount of control messages produced during the discovery process. The aforementioned results highlight the main outcome of this paper: switching between the proposed topology control strategies based on information handled by the software defined controller, i.e., the detected network conditions, topology and application requirements, can improve SDWSNs performance.

VI. Conclusions

In this paper, we addressed issues related to topology control for SDWSNs. Topology control is a key factor in WSNs responsible for the seamless, smooth and efficient operation of the network. Our experimental results show that software defined techniques can improve the topology control in WSNs, while providing robust results. The proposed algorithms are reducing the initial latency caused by the need of communication with a controller, and subsequently improve the time to establish data flows in the network, compensating even the increased control traffic introduced by the SDN paradigm to WSNs. As a future work, we are planning to include mobility and heterogeneous environment scenarios, investigating in depth topology maintenance issues as well. We also plan to conduct experiments increasing the number of nodes in the network in real testbeds, i.e., using both the WiSHFUL [22] and MONROE [23], [24] novel experimentation platforms. Last but not least, an important step forward is to introduce relevant topology control strategies for the evolution of OpenFlow beyond infrastructure networks.

Acknowledgment

This work is partially supported by the Horizon 2020 NECOS (grant agreement no. 777067) project and the open call schemes of the WiSHFUL (grant agreement no. 645274) and MONROE (grant agreement no. 644399) projects.

References

[1] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. John Wiley & Sons, 2010.

[2] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey," in *19th Int. Conf. on Software, Telecommun. and Comput. Networks (SoftCOM)*, 2011.

[3] I. T. Haque and N. Abu-Ghazaleh, "Wireless Software Defined Networking: A Survey and Taxonomy," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 4, pp. 2713–2737, 2016.

[4] P. Santi, "Topology Control in Wireless Ad Hoc and Sensor Networks," *ACM Comput Surv*, vol. 37, no. 2, pp. 164–194, Jun. 2005.

[5] M. Li, Z. Li, and A. V. Vasilakos, "A Survey on Topology Control in Wireless Sensor Networks: Taxonomy, Comparative Study, and Open Issues," *Proc. IEEE*, vol. 101, no. 12, pp. 2538–2557, Dec. 2013.

[6] G. Violettas, T. Theodorou, S. Petridou, A. Tsioukas, and L. Mamatas, "An Experimentation Facility Enabling Flexible Network Control for the Internet of Things," in *Proc. of the IEEE Conf. on Comput. Commun. (INFOCOM)*, Atlanta, 2017.

[7] X.-Y. Li, Y. Wang, and W.-Z. Song, "Applications of k-local MST for topology control and broadcasting in wireless ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 12, pp. 1057–1069, Dec. 2004.

[8] Z. Huang, C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Topology control for ad hoc networks with directional antennas," in *Proc. of the 11th Int. Conf. on Comput. Commun. and Networks*, 2002, pp.16-21.

[9] S. A. Borbash and E. H. Jennings, "Distributed topology control algorithm for multihop wireless networks," in *Proc. of the Int. Joint Conf. on Neural Networks, 2002,* vol. 1, pp. 355–360.

[10] D. M. Blough, M. Leoncini, G. Resta, and P. Santi, "The k-Neighbors Approach to Interference Bounded and Symmetric Topology Control in Ad Hoc Networks," *IEEE Trans. Mob. Comput.*, vol. 5, no. 9, pp. 1267–1282, 2006.

[11] R. Wattenhofer and A. Zollinger, "XTC: a practical topology control algorithm for ad-hoc networks," in *Proc. of the 18th Int. Parallel and Distributed Processing Symposium,* 2004.

[12] J. Vasseur, N. Agarwal, J. Hui, Z. Shelby, P. Bertrand, and C. Chauvenet, "RPL: The IP routing protocol designed for low power and lossy networks," *Internet Protoc. Smart Objects IPSO Alliance*, vol. 36, 2011.

[13] L. Ochoa Aday, C. Cervelló Pastor, and A. Fernández Fernández, "Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks," External research report, 2015.

[14] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in OpenFlow-based Software Defined Networks," *Comput. Commun.*, vol. 77, pp. 52–61, Mar. 2016.

[15] John Kaippallimalil, "Open Networking Foundation Wireless & Mobile Working Group." [Online]. Available: https://www.opennetworking.org/.

[16] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[17] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *27th Biennial Symp. on Commun. (QBSC)*, 2014, pp. 71–75.

[18] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software Defined Wireless Networks: Unbridling SDNs," in *European Workshop on Software Defined Networking (EWSDN)*, 2012, pp. 1–6.

[19] 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 *2015 IEEE Conf. on Comput. Commun. (INFOCOM)*, 2015, pp. 513–521.

[20] B. Oliveira, C. Borges Margi, and L. Batista Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *2014 IEEE Latin-America Conf. on Commun. (LATINCOM)*, 2014, pp. 1–6.

[21] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined Wireless Sensor Networks," in *2016 IEEE Int. Symp. on Consumer Electronics (ISCE)*, 2016, pp. 85–86.

[22] C. Fortuna *et al.*, "Wireless software and hardware platforms for flexible and unified radio and network control," in *European Conf. on Networks and Commun.*, 2015, pp. 712–717.

[23] Ö. Alay *et al.*, "Measuring and assessing mobile broadband networks with MONROE," in *2016 IEEE 17th Int. Symp. on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2016, pp. 1–3.

[24] Alay, Ozgu, *et al.* "MONROE: Measuring mobile broadband networks in Europe." in *Proc. of the IRTF & ISOC Workshop on Research and Applications of Internet Measurements (RAIM)*. 2015.