Performance Analysis of Software Defined Network Concepts in Networked Embedded Systems

Bach Tran, Mohamed Elamin, Nghi Tran(✉), and Shivakumar Sastry

Department of Electrical and Computer Engineering, University of Akron, Akron, OH 44325-3904, USA
{bxt1,mae72}@zips.uakron.edu, {nghi.tran,ssastry}@uakron.edu

Abstract. We present a novel approach to incorporating Software-Defined Networking (SDN) in networked embedded systems where the nodes have limited capabilities and the wireless links have limited bandwidth. The SDN controller was implemented on a Beagle Black board. The approach was validated through simulations and experiments on a physical testbed.

Keywords: Wireless-SDN · Networked embedded system · Control overhead · Performance analysis

1 Introduction

Networking and connectivity is an essential infrastructure for our economies, industries, societies and systems and have a central role in managing costs and productivity [1], and our health and wellness [2]. Modern networks confront a few challenges such as complexity and operational costs as the size of the applications increase [3]. The introduction of low-power, resource constrained nodes that are cost-effective for data gathering has pushed the networking issues to their limits because of the tight bandwidth available and the unreliability of the links. Such systems are asynchronous, decentralized and severely limited, and yet they must provide reliable and trustworthy service in a variety of application settings.

SDN helps to address some of these drawbacks by decoupling the control and data planes. Network intelligence and state are logically centralized and the underlying network infrastructure is abstracted from the applications [4]. Users are able to programmatically change the capabilities of the underlying network with minimal disruptions to the applications that rely on the networking infrastructure. A new abstraction layer is added between applications and router/switches. SDN was primarily designed for wired networks with point-to-point (P2P) connections between their nodes [3].

Many emerging networked systems such as Wireless Sensor Networks (WSN) [5,6], Internet of Things [7] and Advanced Manufacturing [8–10] rely on wireless

T. Q. Duong et al. (Eds.): INISCOM 2019, LNICST 293, pp. 99–108, 2019.
https://doi.org/10.1007/978-3-030-30149-1_8

links and have motivated the exploration of SDN services to programmatically alter the capabilities of networked embedded systems. Several challenges must be addressed to achieve this vision. The function of nodes in networked embedded systems is more than just sensing. Although the node has limited resources and computation power, it can still drive basic operations and control heavy machinery. Unlike the fully capable desktop systems used in common networking scenarios, the nodes have limited resources. The radio transceivers used in such systems have limited capabilities and, typically, infrastructure such as cell towers and WiFi are not available. The systems are expected to be deployed in ad-hoc manners and must support incremental growth and change. While the communications requirements in these systems usually follow a set of paths, there would be changes from time to time that require programmatic changes in the network capabilities.

In this paper, we present an approach for adapting key SDN concepts in wireless networked embedded systems. Starting with a well-known SDN controller called SDN-Wise [11] that was recently developed for the Raspberry PI, the controller was adapted to address the needs of networked embedded systems. A finite state machine was designed to execute on commercial mote platforms. The approach was validated both in simulation settings and through physical experiments.

The remainder of this paper is organized as follows: Sect. 2 reviews background in SDN and WSN. Section 3 describes the system's structure and role. Experimental results that demonstrate our system are discussed in Sect. 4, and conclusions are presented in Sect. 5.

2 Background

The section presents some of the approaches used to design and analyze SDN. Some of the key concepts of WSN and protocols proposed are reviewed.

Currently, networks support complex tasks using low-level network configuration commands. The complexity of configuring the systems and managing networking operations increase both as the number of nodes increase and as the communication demand increases. Such low-level commands are not suitable for continually changing environments. Moreover, system reconfiguration is done manually and on a per-case basis and this situation may lead to inconsistency in the network state [12]. Many of these issues can be addressed by using a new architecture that accounts for dynamic changes in data, computation and storage [13].

The key idea in SDN is to separate or decouple the control plane and the data plane. This idea was introduced in other systems and protocols. For example, in [14], the authors introduced a system that mimics forwarding decisions, made by regular hardware-based switches, in software and applied these rules for new packets. Their goal was to increase the system's functionality space without changing the underlying hardware. Thus, they made the network improvements cycle much faster. The 4D environment described in [15] divided the process into four planes: decision, dissemination, discovery and data. While all the decisions are made in the decisions plane using an autonomous system and decision

elements, the data plane was responsible for only forwarding packets depending on the decision plane's output. That resulted in a system where controlling and forwarding are prevented from running on the same network element and gave an extra level of abstraction and simplicity. Several efforts to standardize the core networking ideas have helped to mature this idea of separating the control and data planes [16–18].

The controller is the central artifact of the control plane. This plane is logically centralized and is essential to differentiate between the operations needed to manage networking capabilities and the operations necessary to forward data packets between the nodes.

A single, centralized, controller is simple to design and implement. Such an architecture reduces the chances for logical inconsistencies. However, such a node has a high risk and is a single-point of failure. It suffers from scalability limitations because when the network starts growing, the computation power of the controller must grow at a higher rate to ensure the network's performance [19]. In an effort to target the drawbacks of centralized controllers the authors in [20] introduced the idea of decentralizing SDN's control plane both physically and logically by defining the concepts of a control hierarchy which consisted of main and secondary controllers. This helped the network to distribute the load among different physical controllers thus avoiding network's bottlenecks; and reduce the cost of needing high computation power. Many other architectures were proposed [21–23]. Nevertheless, these techniques increase the difficulty of maintaining a consistent network view and network decisions.

The main role of a flow-table is to represent the multi-hop connectivity structure between nodes. The flow-table has all the characteristics of a routing-table but there are two major differences. First, flow-tables are only updated with information coming from the controller. Second, while routing-tables are generally stateless, a flow-table can be designed to make decisions based on its state and change specific attributes in the packet.

The configuration commands that the controller sends are used to update flow-tables inside each SDN node. Each table contains *flow entries*, each with three sections. The *Matching* section has the rules and attributes that must be compared and checked for each incoming packet. The *Action* section specifies what must be done for each matched packet. The *Statistical Information* section gathers information about the flow-table entry [11,24]. Generally, a packet can match more than one rule and the relevant actions will determine its flow through the system. Unlike simulated wireless links, the physical wireless links present many challenges that affect the performance of the network. Although finding the shortest path is preferable in most cases, the throughput of the network is affected by traffic congestion in specific nodes, limited capabilities in the node, and the effects of hidden/exposed terminals. Thus, we may increase the network's throughput by choosing other paths in the network [25].

Many estimation techniques were proposed to target specific features such as rapid calculation, memory efficiency and performance by using information from the physical layer, link layer or the network layer. The physical layer can indi-

cate the channel quality and the errors in the packets, but it will not take into consideration lost packets. The link layer can measure the delivery of the sent packets through acknowledgments. The network layer can indicate the importance of each link and how to share node's estimations with other nodes [26]. The authors in [25] introduced an estimator that depends on the link layer's information. They would calculate the number of transmissions required for a packet to successfully be sent plus the number acknowledgments received and then calculate the delivery ratio for each link. The HyperLQI [27] uses the success rate maintained by link layer acknowledgments in addition to the channel Link Quality Indicator (LQI), from the physical layer to give an estimation of each link. A link quality estimator that uses information from all the three networking layers is presented in [26].

3 System Structure

This section presents the versatile design of the physical Wireless-SDN node that can support the deployment of multiple network topologies, such as tree, ring and mesh. The CSMA/CA protocol B-MAC [28] was used in all the nodes. The IRIS from MEMSIC motes used as the platform for experiments consist of an ATmel Atmega1281, a low-power 8-bit microcontroller, with 128KB in-system programmable flash memory and 8KB of RAM. Each node includes an ATmel AT86RF230 wireless transceiver, which is a 2.4 GHz ZigBee IEEE 802.15.4 compliant module. The RF module has a maximum data rate of 250Kbps, an automatic reception acknowledgment feature and power detection [29].

SINK Node. The SINK node, which is connected to the controller through a serial (USB) connection, serves as the interface between the control and data planes. All the packets from every node to the controller, and vice versa, must go through the SINK. The SINK reorganizes those packets here so that the controller can handle. In addition, this node is responsible for initiating the topology formation. When the SINK powers-up, it send a PROXY packet to the controller to introduce its self as a valid and ready SINK node. After that, the SINK will start the topology discovery protocol by propagating BEACON packets periodically.

Beaconing and Topology Discovery. Every node must maintain a valid route to the SINK and, thus, the controller. The BEACON packets and the topology discovery protocol are designed to help in this task. Except for the SINK, every node will be powered up to idle state, where it listen to the channel waiting for a BEACON packet to be broadcast by any other neighbor.

Routes to the SINK. The BEACON packets contain the sender's distance from the SINK and its battery level. On the first receiving of this packet, a node will change its state to ACTIVE, register the sender as next hop to the SINK

and start all the timers within it. When an ACTIVE node receives a BEACON packet, it will compare its distance to the SINK with the sender's distance and measures the quality of that link using the estimation. If both appear to be better, the node will update its route to the SINK based on the new one, otherwise it will just update its neighbor-table with this sender.

Neighbor-Table. Each node maintains a set of single hop's neighbors that can be directly reached, and keeps track of each neighbor in terms of TTLs and link quality. This process is memory intensive and must be managed carefully. A large number of neighbors will result in a strongly connected graph with short mean distance to each destination, but it will affect the performance of the nodes, increase search time, and cause a larger number of REPORT packets to be generated. On the other hand, a smaller neighborhood will result in longer paths and a higher chance of the node being unreachable.

Reports to Controller. The REPORT packets periodically inform the controller about the neighborhood of each node in the system.These packets contain a list of all the node's neighbors, available battery levels, distance from SINK and quality of each link.

Controller. The controller logic used in our experiments is studied in [11]. This controller uses the weighted version of Dijkstra algorithm to find the best route between two points using both the number of hops and link quality. It maintains a map of the network depending on both the REPORT packets received and an internal expiration timer. The system takes into account that not every link is a bidirectional one unless it was reported by both nodes that are at either end of the link. When a controller receives a REQUEST packet from the SINK, it will extract the source and destination information. Based on the latest map it executes the Dijkstra algorithm to find the shortest path between the nodes. The controller will configure all the nodes in the path by generating a single OpenPath packet that contains all their address to the final destination.

Time to Live and Flow-Table Entries. Because of the limited memory space and the goal of reducing searching time, each node will keep track of a certain number of the most active neighbors. The validation time associated with each entry in the Flow-table and neighbor-table is updated in every TTL's period. If the initial value is very large, the node will exhaust its resources on non-useful entries and will keep track of non-existent neighbors. On the other hand, if this value is too small, the network will suffer from a huge overhead of sent REQUEST packets and the node will not have a stable neighbor list. When the TTL timer triggers, the node will go through the Flow-table and the neighbor-table and remove all entries with TTL values less than the TTL period. A neighbor's TTL is reset when the node hears a packet generated from that specific neighbor, and a flow-table entry's TTL is reset only if that entry is used.

Wireless Link Estimator. In our topologies, we used a link quality estimator that collects information from physical and link layers, depending on both broadcast and unicast transmissions to give the overall evaluation. Each estimate is updated after a predefined number of packets. The node will evaluate the broadcast packets by detecting their energy on reception and comparing it to a threshold value. Unicast messages were measured by counting the number of sent and acknowledged packets on every link to calculating the success and failure rates. To avoid fast changing environment, we assigned weights to the new estimate New and the previous estimate $Estimation_{old}$ using the exponential moving average window as:

$$Estimation_{final} = \alpha New + (1 - \alpha) Estimation_{old}.$$

here α denotes the estimation factor. This average estimates are calculated separately for broadcast and unicast transmissions as follows:

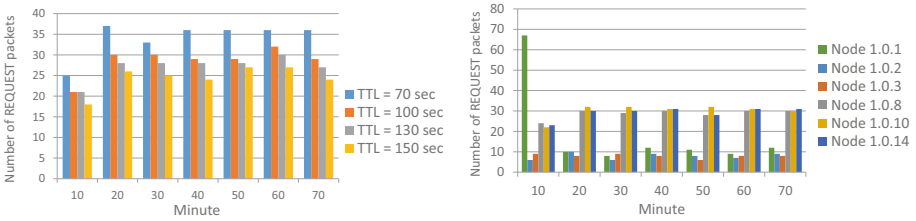
- Broadcast
  1. After receiving a specified number of broadcasts from a source, the node calculates an immediate estimate.
  2. This value is combined with the previous broadcast-estimate values.
  3. At the end of broadcast estimation we obtain $Estimation_{final}$.
- Unicast
  1. After sending the specified number of unicast messages over a specific link, the node will keep track of the number of acknowledged and unacknowledged packets.
  2. The success rate is then calculated.
  3. The success rate will be used as the new estimate to find $Estimation_{final}$.

The estimator keeps track of a number of links that are not part of the neighbor-table. These are important because they will serve as a base to compare neighbor's links. At the end of each TTL cycle, the node will check all the available links in the table to figure out the best value and all the links of the neighbor will be compared to that value. In this way, the node only keeps the most reliable links in the neighbor-table.

4 Results

This section presents the simulation and experiment results obtained from the network with a collection of motes. The observed behavior of the network provides understanding on which the system should be designed in order to minimize the interaction between the control and data planes.

Simulation Approach. Two network topologies were designed to understand the baseline behavior of the Wireless-SDN network, one is a 16 node mesh and the other is a 14 node tree network, both with a single SINK labeled 1.0.1. We use the mesh topology to examine the effect of the time-to-live parameter and



(a) (b)

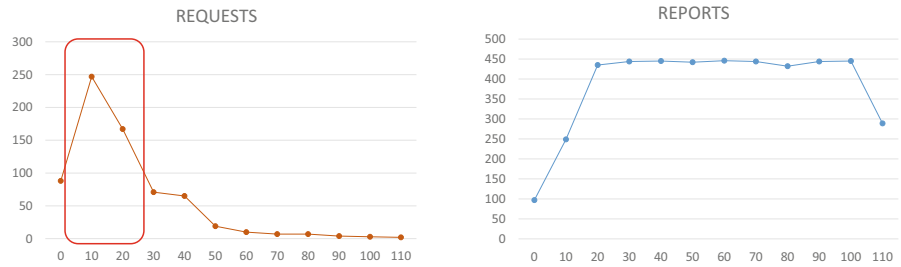
Fig. 1. Average number of REQUEST packets generated by an edge node in a mesh topology (a) and those generated by different nodes (intermediate and leaf nodes) in a tree topology (b).

Table 1. Major differences between simulated systems and physical systems.

	Simulation environment	Physical system
Links	Static	Dynamic
Link quality	Fixed	Changing
Transmission	P2P	Inherently broadcast
Memory	Virtually unlimited	Limited
Processing time	Negligible	Significant

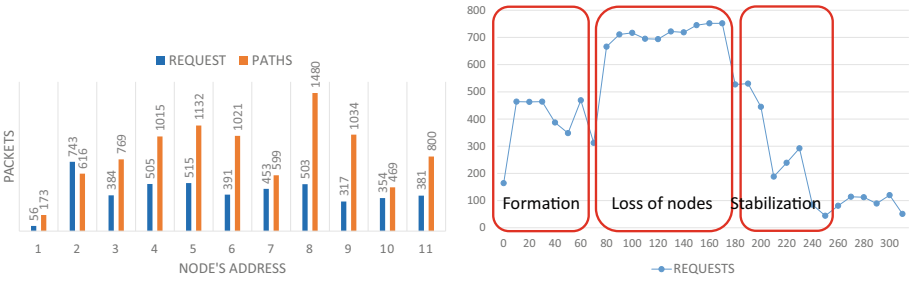
the tree topology to examine the control overhead of a node with respect to its position. Figure 1a shows the average number of REQUEST packets generated from the edge node in a mesh topology for every 10 min. With lower value of TTL, each flow entry has a shorter time until it is removed from the flow table, therefore the node then need to send REQUEST packet to the SDN controller more frequently in order to update the expired flow entry. Figure 1b shows the difference between the number of REQUEST packets generated from different nodes in the tree topology for every 10 min. Node 1.0.2 and 1.0.3, which are the immediate child nodes of the root, have to generate less REQUEST packets than the leave node 1.0.8, 1.0.10, and 1.0.14. The main reason is that the intermediate nodes can update their flow-tables without generating additional REQUEST packets based on the OpenPath generated by the controller.

Experiments With IRIS Motes. The physical implementation of a Wireless-SDN using commercially available mote platforms presents a few additional challenges in addition to what can be realized in simulation environments, which are summarized in Table 1. Figure 2a clearly shows that the network goes through different phases. The highlighted section in this figure corresponds to the formation phase where the REQUEST packets are rapidly generated. A steady level of reporting will allow the SDN controller to beresponsive to new REQUEST packets. After the flow-tables are configured, the number of REQUEST and REPORT packets remains stable during the lifetime of the network. A compar-



(a) (b)

Fig. 2. The overhead in an SDN network comprises of REQUEST packets and REPORT packets.



(a) (b)

Fig. 3. A comparison between active and passive learning in topology discovery on the left (a) and three phases of a large network on the right (b).

ison between the number of REQUEST packets generated and the OpenPath packets received is presented Fig. 3a. Notice that the nodes that receive more OpenPath packets tend to generate fewer REQUEST packets. This result indicates the importance of node's position in the topology, i.e., the number of paths on which the node lies. It give the designers a suggestion of critical point of failure and load balancing algorithms. The system was tested in a larger network with 18-nodes and high packet generation rates, and its behavior is shown in Fig. 3b. The first section in this figure presents the number of REQUEST packets during the formation phase. In the second section, the network suffered from the loss of four nodes that led to the increase in Control overhead, and in the final one the network recovered as the depleted nodes re-connected to the topology.

5 Conclusion

This paper presented the design and implementation of a Wireless-Software Defined Network that is suitable for a variety of networked embedded systems. The design was validated in a simulation setting and through experiments using

commercial motes. The system model can be adapted to different hardware platforms because the design relies on common features that are available in a variety of platforms. This paper presented the general behavior and key parameters to analyze the network performance in typical operational scenarios. Insights into these features allow us to have more control on the flow of packets through a network both at the system level and at the level of individual nodes. The results are encouraging and serve as a foundation for several related investigations in the future.

Acknowledgement. This material is based upon work supported by the US National Science Foundation (NSF) under Grant 1531070. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Goel, R., Mehta, A., Dua, S.: Next generation networks: regulation and interconnection in modern communications era. In: Networks 2006, 12th International Telecommunications Network Strategy and Planning Symposium, pp. 1–5, November 2006
2. Lin, Y.-J., Lai, J.-S., Wu, Y.-J., Ou, M.-J.: A communication system for living alone elders. In: HEALTHCOM 2006 8th International Conference on e-Health Networking, Applications and Services, pp. 108–113, August 2006
3. Sezer, S., et al.: Are we ready for SDN? implementation challenges for software-defined networks. IEEE Commun. Mag. **51**, 36–43 (2013)
4. ONF, "The new norm for networks," white paper (2012). https://www.opennetworking.org
5. Dargie, W., Poellabauer, C.: Fundamentals of Wireless Sensor Networks: Theory and Practice. Wiley Publishing (2010)
6. Burri, N., von Rickenbach, P., Wattenhofer, R.: Dozer: ultra-low power data gathering in sensor networks. In: 2007 6th International Symposium on Information Processing in Sensor Networks, pp. 450–459, April 2007
7. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
8. Archer, B., Sastry, S., Rowe, A., Rajkumar, R.: Profiling primitives of networked embedded automation. In: 2009 IEEE International Conference on Automation Science and Engineering, pp. 531–536, August 2009
9. Mahamadi, A., Chippa, M., Sastry, S.: Reservation based protocol for resolving priority inversions in composable conveyor systems. J. Syst. Architect. **74**, 14–29 (2017)
10. Hayslip, N., Sastry, S., Gerhardt, J.S.: Networked embedded automation. Assembly Autom. **26**(3), 235–241 (2006)
11. Galluccio, L., Milardo, S., Morabito, G., Palazzo, S.: SDN-wise: design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 513–521, April 2015
12. Kim, H., Feamster, N.: Improving network management with software defined networking. IEEE Commun. Mag. **51**, 114–119 (2013)

13. Metzger, A., Cassales Marquezan, C.: Future Internet apps: the next wave of adaptive service-oriented systems? In: Abramowicz, W., Llorente, I.M., Surridge, M., Zisman, A., Vayssière, J. (eds.) ServiceWave 2011. LNCS, vol. 6994, pp. 230–241. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24755-2_22
14. Casado, M., Koponen, T., Moon, D., Shenker, S.: Rethinking packet forwarding hardware. In: HotNets, pp. 1–6 (2008)
15. Greenberg, A., et al.: A clean slate 4D approach to network control and management. ACM SIGCOMM Comput. Commun. Rev. **35**(5), 41–54 (2005)
16. Berman, M., et al.: GENI: a federated testbed for innovative network experiments. Comput. Netw. **61**, 5–23 (2014). Special issue on Future Internet Testbeds - Part I
17. ONF Overview. https://www.opennetworking.org/about/onf-overview. Accessed 30 Mar 2017
18. Linux Foundation: Opendaylight - an open source community and meritocracy for software defined networking, A Linux Foundation Collaborative Project
19. Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T.: Software-defined networking: challenges and research opportunities for future Internet. Comput. Netw. **75**, 453–471 (2014). Part A
20. Santos, M.A.S., Nunes, B.A.A., Obraczka, K., Turletti, T., de Oliveira, B.T., Margi, C.B.: Decentralizing SDN's control plane. In: 39th Annual IEEE Conference on Local Computer Networks, pp. 402–405, September 2014
21. Ng, E., Cai, Z., Cox, A.: Maestro: a system for scalable openflow control, Rice University, Houston, TX, USA, TSEN Maestro-Technical report, TR10-08 (2010)
22. Hassas Yeganeh, S., Ganjali, Y.: Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 19–24. ACM (2012)
23. Tootoonchian, A., Ganjali, Y.: Hyperflow: a distributed control plane for openflow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, p. 3 (2010)
24. Openflow® switch specification ver 1.5.1. https://www.opennetworking.org/sdn-resources/technical-library April 2015. Accessed 11 Apr 2017
25. De Couto, D.S., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. Wirel. Netw. **11**(4), 419–434 (2005)
26. Fonseca, R., Gnawali, O., Jamieson, K., Levis, P.: Four-bit wireless link estimation. In: HotNets (2007)
27. Gupta, A., Sharma, M., Marot, M., Becker, M.: HybridLQI: hybrid multihopLQI for improving asymmetric links in wireless sensor networks. In: 2010 Sixth Advanced International Conference on Telecommunications, pp. 298–305, May 2010
28. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 95–107, ACM (2004)
29. Shukri, S., et al.: Analysis of RSSI-based DFL for human detection in indoor environment using IRIS mote. In: 2016 3rd International Conference on Electronic Design (ICED), pp. 216–221, August 2016