Using Software-defined Networking Principles for Wireless Sensor Networks

Martin Jacobsson and Charalampos Orfanidis
Uppsala University
Department of Information Technology
Email:{martin.jacobsson, charalampos.orfanidis}@it.uu.se

***Abstract*—In this paper, we propose an architecture based on software-defined networking (SDN) for wireless sensor networks. Ideas of how to design and make use of the flexibility that SDN offers are presented. We discuss how SDN principles can lead to the use of commodity hardware in a wider range of WSN deployments and then tailor the software only to meet the requirements of the specific deployments and their applications. A few examples are introduced that demonstrate how the architecture can be used for networking, in-network processing, and performance predictions.**

I. Introduction

More and more physical objects are becoming smart and connected using ICT. A key concept for this is wireless sensor networks (WSN), which connects smart objects with each other in a local area using low power wireless communications. WSN is an important building block for the future Internet of Things (IoT) vision.

WSNs may be used for many different types of applications and in many different types of environments, such as in residential homes, apartment buildings, large office buildings, industrial plants, outdoor in urban environments as well as rural areas. Applications may be delay-tolerant collection of sensor readings, responsive smart home applications, tracking of mobile objects or persons, and time- and mission-critical operation of industrial plants, to name a few.

For the IoT vision to become true, it is important to keep the costs low and this is mainly achieved through economy of scale in both hardware and software. This means both standardized hardware and software. At the same time, application requirements may force us to tailor every single WSN deployment individually and this can only be achieved by a high degree of reconfiguration capabilities, something that can be realised by using concepts and ideas from software-defined networking (SDN) [1].

The developments in hardware have made reconfiguration possible. The TelosB WSN node platform [2], which is the most commonly used experimention platform in the research community, is now 10 years old. Today, for example, ARM and its partners have released new energy-efficient microcontrollers, such as the ARM Cortex M-series, which are suitable for WSN nodes. Some of these have a 32-bit architecture and much more program and code memory as well as faster processing capabilities than TelosB. Also the wireless chips have been updated. All this enables much more functions in software.

This extra functionality enables a much higher degree of tailoring, but also more advanced in-network processing. The latter also provides shorter communication paths between sensors, actuators, and the data processing. This also leads to more robustness due to less dependence on far away nodes, as well as lower delays and less energy consumption due to the minimized communication paths.

In this paper, we discuss how SDN concepts can lead to the use of commodity hardware in a wider range of IoT and WSN applications and then tailor-making using software to meet the requirements of the specific deployments.

The remainder of this paper is organised as follows. Section II introduces the SDN-based architecture for WSNs. In Section III, we discuss the networking aspects of the architecture, while in Section IV, we give some example SDN applications. Section V contains the related work and Section VI concludes the paper.

II. An SDN Architecture for WSN

SDN is a concept developed to meet the demands of more flexibility in the networking implementations on Internet routers. In SDN, there is a decoupling defined between the control plane and the data plane with OpenFlow [3] being the currently most successful standard. New network control and management solutions can easily be deployed by replacing the control plane functionality.

OpenFlow is based on TCP, which means that the control function could run locally on the router or somewhere else, such as at a central server. Through the interface, the controller application can configure the forwarding tables of the router (or network switch). The task of the controller is to provide a coherent image of the entire network and provide it as one single entity towards the SDN applications. Typical SDN applications could be routing, but also other functions, such as access control and software-based traffic analysis, are possible.

The aim of this paper is to propose a flexible architecture for WSN and IoT systems based on ideas from SDN and where in-network processing is a natural part. We have to mention that an SDN architecture for WSNs includes challenges. The WSNs have constrained resources and most of the times are battery operated, which means that the available energy should be managed efficiently. One of the basic functions in a SDN architecture is the communication between the control and data plane and an increase in the communication will increase

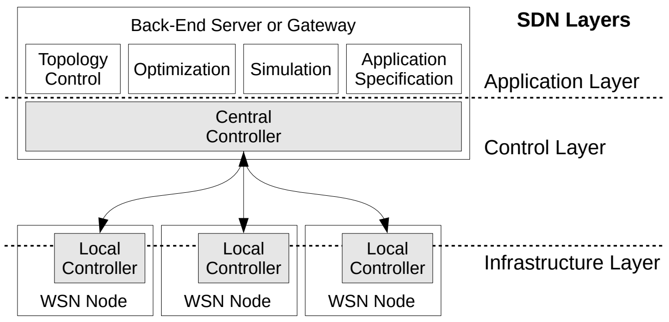

Fig. 1. Overall SDN architecture for a WSN

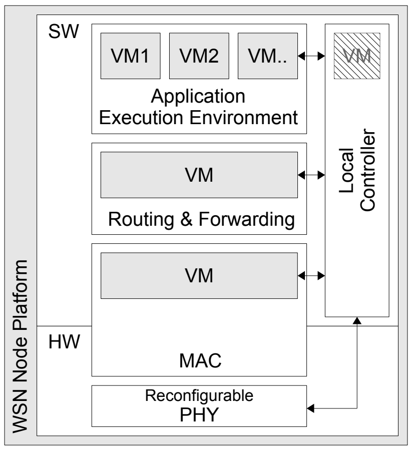

Fig. 2. The architecture of the WSN node with SDN support

the energy consumption. Hence, the energy efficiency is one factor that should be considered during the design of a WSN architecture based on SDN.

Figure 1 shows the overall architecture of our proposal with the SDN layers indicated. Each WSN node is equipped with a local controller, whose functionality can be as simple as just receiving and executing the commands from the central controller. The central controller communicates over the network with either the used routing protocol if it is running or simple networking principles, such as network-wide flooding [4]. On top of this, message formats between the controller and the WSN nodes must be defined.

On top of the central controller, there are one or more SDN applications. These applications can be related purely to the networking of the WSN, such as topology control and routing. In this architecture, some SDN applications may be directed towards the network operator staff with a user interface, while others are completely automated. SDN applications may use optimization solvers, simulators, specifically made algorithms, or a combination.

Figure 2 shows the functionality on the WSN Nodes. Everything, except potentially some parts of the local controller resides in the infrastructure layer. The main task of the local controller is to setup, reconfigure, and monitor the parts of the software that can be reconfigured. It can do so in two ways. Either it changes parameters in the different functions, such as the central frequency of the radio, the retransmission limit in the MAC layer, modifying entries in the forwarding table, etc., or it installs new code in the different functions that changes the behavior. The latter can be done by virtual machines (VM), but can also be done with native code and dynamically linked library functions. In this way, not only routing and MAC can be modified by the controller, but there is also flexibility for the neighbour and topology discovery functionality as well as other functions.

In-network processing of sensor data is important for the robustness of applications, energy-efficiency, and the reduction of the delay. An application execution environment is defined that allows application code to be updated over the air that can process sensor data as it is being forwarded by the nodes. For instance, the ProFuN TaskGraph Tool [5] can be used.

With a good code execution environment in place on the WSN nodes, it also becomes possible to distribute the controller function. I.e., it is no longer required to run the controller on a central node. Furthermore, hybrid controller versions can be defined.

In homogenous networks, native binaries and dynamic linking can be used, while massively heterogenous networks may benefit from using byte code and VM technology designed for embedded systems. Contiki OS has good support for dynamic loading as well as over the air programming, which can be used if all run the same software and micro-controller architecture. If various hardware and software systems exist in the same network, VMs may be a better choice. Some researchers have proposed to put a virtual machine (VM) on WSN nodes. In this way, one single byte code binary can be distributed and executed on any node. Examples of VMs developed for WSN nodes include EmbedVM, TakaTuka [6], and Darjeeling [7]. The latter two are Java VMs. A third option is using something similar to the command chains of snapMac [8], i.e., a domain-specific VM-like engine with a small set of pre-defined commands that can be executed by a very simple byte code interpreter. A fourth option is to use very simple scripting languages. Whatever choice is made, it is also important to have a good run-time monitoring system that, for instance, can deal with application processes running out of hand or debugging. VMs and scripting languages may be better in offering that functionality.

III. Reconfigurable Networking

With SDN, we have the option to centralize all or some parts of the networking at a more powerful node. For robustness reasons, we may not centralize everything. Instead, MAC, forwarding, and many routing decisions are still likely to be carried out by the individual nodes. However, long-term decisions can be taken by a centralized controller, such as which protocol and what parameters to use. A central controller may also have good knowledge about application requirements, which can be taken into account as well.

The central controller needs to discover the actual topology and the quality of the links. This can be done either by packet trace information or simpler link quality estimation (LQE) for the links of the network. What is best depends on how it is supposed to be used. Hence, even this part should be reconfigurable. Only when data is needed by an SDN application, the collection of data should be started, since every extra data collection will reduce the network life-time.

A packet trace contains detailed information over how packet losses over a link behaves. This information can be used in a simulator (e.g., [9]) with a technique known as trace-based simulation. The simulator can be used to answer questions, such as which protocol is best or which parameters should be selected. The good thing is that the accuracy between the simulator and the deployment will be very good since the trace is collected from the network under study.

While trace-based simulations can offer very good accuracy, trace data is expensive to collect. Hence, more light-weight alternatives are needed. LQE [10] is used to predict a link's quality for various reasons, such as transmission power control, rate adaptation, and routing. Many such techniques have been proposed [10] and in general they are designed to be light-weight. The collection of LQE for all links and thereby also the network topology has been discussed by a handful of papers, such as [11]–[13].

To gather LQE information from all links in the network and model this for use by the SDN applications is not a trivial problem. There are many aspects that need to be looked at and that have implications on the overall performance. Questions to be answered includes: What data to collect and how? Packet reception ratio (PRR), received signal strength (RSS), signal-to-noise ratio (SNR), noise level, chip/bit error rate, etc. How often to sample and collect topology information? How much energy do we consume by doing these measurements? Do we gain by doing it? What happens to the energy efficiency by introducing all this extra overhead?

Many of the answers depend on how the data is being used. The LQE method may also have to take into account different packet sizes, transmission powers, and data rates, depending on the underlying wireless technology being used. We need to define light-weight mechanisms for collecting topology information that is still useful. Since this functionality also has to evolve, we need this collection mechanism to also be replaceable. By updating the code, what is collected, how it is processed, and when it is communicated can be changed to meet future requirements.

IV. Further SDN Applications

When the topology and link information has been collected, it can be used for various reasons. Here, we apply this information in four more ways.

A. Centralized Networking

If LQE or packet trace information is available at the controller, we can use it for routing or tuning routing parameters. In [13], one approach is attempted for centralized routing. The tuning of parameters based on information about the network has been shown by [8], [11]. In general, simulation or optimization solvers may be used to find the right configuration. The new thing here is that we can also take application needs into account in this process.

B. Optimal code deployment

Depending on the topology, where a piece of code is executed may have importance on the network life-time and robustness. If the processing of sensor data is done on one of the intermediate nodes, data does not need to make a detour to be processed. Good data about the network can be used together with a solver to find the optimal assignment [5], [14].

C. Predicting a WSN networks behavior and performance

Given the topology information, the used protocols, the application behavior, and everything else, it becomes possible to simulate the actual network at hand and predict its behavior and performance. For instance, what is the actual network life-time likely to be and what protocol is best for this particular network and application? Robustness analysis can also be done by simulating link or node outages. A system could also be envisioned that goes one step longer and suggests the network operator to make alterations to the node deployment, such as move or add nodes, use of other antennas, etc.

D. Network life-time prediction

Using good energy consumption models [15] together with collected network data, such as radio propagation information and battery levels, better predictions of the network life-time can be achieved. For the overhearing and collisions, it may also be necessary to collect interference information. Interference can be collected by the WSN nodes in a similar fashion to JamLab [16].

V. Related Work

Early on in the WSN research, it has been noted that some MAC protocols work better in some scenarios and other protocols in other scenarios. Hence, reconfigurable or adaptive MAC platform have been proposed. T-MAC [17] is one such early protocol where the listening period is adaptive. C-MAC [18] is a much more recent protocol where many parameters can be reconfigured by the applications. In pTunes [11], the authors treat the parameters of a MAC protocol as an optimization problem. They collect data from the network at the base station where a solver is used to find the optimal parameter configuration. The network is then reconfigured accordingly. With snapMac [8], it is possible to program the radio layer with a sequence (called chain) of simple commands. The idea is similar to a VM with an extensible set of very low-level instructions.

IMPERIA [13] is a centralized architecture for WSNs with data gathering applications. It defines MAC, routing, and management protocols. Its main idea is to move functionality from the simple WSN nodes to the sink (i.e., the controller functionality) in a similar way to a SDN. However, the flexibility of the SDN concept is not at all studied.

Using SDN in WSNs has been proposed before. Mahmud et al. [19] propose to deploy OpenFlow in WSNs, but the contributions are limited. The same holds for Luo et al. [20]. In [12], Costanzo et al. propose an architecture for a SDN-based WSN where forwarding tables and in-network aggregation of sensor data can be configured. A method for collecting topology information and configuration packets are proposed.

Software-defined radio (SDR) is another related area where radio hardware is replaced with software or reprogrammable hardware (e.g., FPGA) for increased flexibility. The required intensive signal processing is very energy costly [21] and might not be an option for WSN nodes in the foreseeable future.

VI. Conclusions

In this paper, we have proposed a SDN-based architecture for flexible reconfiguration of WSN networking and in-network processing functionality. SDN is an important building block for being able to use standardised low-cost off-the-shelf hardware and yet achieve customization suitable to the individual deployments. We have discussed how the architecture can be used for different purposes, including networking, in-network processing, and WSN management tasks.

An important step in validating the architecture is to implement a prototype version of it. In the future, we aim to make such a prototype and demonstrate its flexibility by implementing multiple controllers and SDN applications on common WSN hardware for different types of applications, such as data collection applications as well as sense-compute-act type of applications. An interesting research question to be addressed in the future is also the investigation of the trade-off between increased SDN functionality and energy efficiency.

Acknowledgments

This research is partly sponsored by the Swedish Foundation for Strategic Research (SSF) under research grant RIT08-0065 for the project ProFuN: A Programming Platform for Future Wireless Sensor Networks.

References

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *Communications Surveys Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[2] J. Polastre, R. Szewczyk, and D. Culler, “Telos: Enabling ultra-low power wireless research,” in *the Fourth International Symposium on Information Processing in Sensor Networks (IPSN’05)*, Los Angeles (CA), USA, Apr. 15, 2005.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: Enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[4] M. Jacobsson, C. Guo, and I. G. Niemegeers, “A flooding protocol for manets with self-pruning and prioritized retransmissions,” in *the International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN’05)*, Washington DC, USA,, Nov. 7-10, Nov. 7-10, 2005.

[5] A. Elsts, F. H. Bijarbooneh, M. Jacobsson, and K. Sagonas, “Demo abstract: ProFuN TG: A tool using abstract task graphs to facilitate the development, deployment and maintenance of wireless sensor network applications,” in *the 12th European Conference on Wireless Sensor Networks (EWSN’15)*, Porto, Portugal, Feb. 9-11, 2015.

[6] F. Aslam, C. Schindelhauer, G. Ernst, D. Spyra, J. Meyer, and M. Zalloom, “Introducing TakaTuka: a java virtualmachine for motes,” in *the 6th ACM conference on Embedded network sensor systems (SenSys’08)*, Raleigh (NC), USA, Nov. 5-7, 2008.

[7] N. Brouwers, K. Langendoen, and P. Corke, “Darjeeling, a feature-rich VM for the resource poor,” in *the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys’09)*, Berkeley (CA), USA, Nov. 4-6, 2009.

[8] P. D. Mil, B. Jooris, L. Tytgat, J. Hoebeke, I. Moerman, and P. Demeester, “snapMac: A generic MAC/PHY architecture enabling flexible MAC design,” *Ad Hoc Networking*, vol. 17, pp. 37–59, Jun. 2014.

[9] A. Marchiori, L. Guo, J. Thomas, and Q. Han, “Realistic performance analysis of wsn protocols through trace based simulation,” in *the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN’10)*, Bodrum, Turkey, Oct. 17-21, Oct. 17-21, 2010.

[10] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, “Radio link quality estimation in wireless sensor networks: A survey,” *ACM Trans. Sen. Netw.*, vol. 8, no. 4, pp. 34:1–34:33, Sep. 2012.

[11] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, “pTunes: Runtime parameter adaptation for low-power MAC protocols,” in *the 11th International Conference on Information Processing in Sensor Networks (IPSN’12)*, Beijing, China, Apr. 16-19, 2012.

[12] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, “Software defined wireless networks: Unbridling SDNs,” in *the 2012 European Workshop on Software Defined Networking (EWSDN’12)*, Darmstadt, Germany, Oct. 2012.

[13] U. Hunkeler, C. Lombriser, H. Truong, and B. Weiss, “A case for centrally controlled wireless sensor networks,” *Elsevier Computer Networks*, vol. 57, no. 6, pp. 1425–1442, Apr. 2013.

[14] F. H. Bijarbooneh and M. Jacobsson, “Macroprogramming of wireless sensor networks using task graphs and constraint solving,” in *the 8th Swedish National Computer Networking Workshop (SNCNW’12)*, Stockholm, Sweden, Jun. 7-8, 2012.

[15] J. Vazifehdan, R. V. Prasad, M. Jacobsson, and I. G. Niemegeers, “An analytical energy consumption model for packet transfer over wireless links,” *Communications Letters, IEEE*, vol. 16, no. 1, pp. 30–33, Jan. 2012.

[16] C. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga, “JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation,” in *the 10th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks (IPSN’11)*, Chicago (IL), USA, Apr. 1214, 2011.

[17] K. L. T. Dam, “An adaptive energy-efficient mac protocol for wireless sensor networks,” in *the 1st International Conference on Embedded Networked Sensor Systems (SenSys’03)*, Los Angeles (CA), USA, Nov. 5-7, 2003.

[18] A. F. R. Steiner, T. Mück, “C-MAC: A configurable medium access control protocol for sensor networks,” in *IEEE Sensors*, Kona (HI), USA, Nov. 1-4, 2010.

[19] A. Mahmud, R. Rahmani, and T. Kanter, “eployment of flow-sensors in internet of things virtualization via OpenFlow,” in *the 3rd FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing (MUSIC’12)*, Vancouver, Canada, Jun. 26-18, 2012.

[20] T. Luo, H.-P. Tan, and T. Quek, “Sensor OpenFlow: Enabling software-defined wireless sensor networks,” *IEEE Communications Letters*, vol. 16, no. 11, 2012.

[21] S. Szilvási, B. Babják, P. Völgyesi, and Á. Lédeczi, “Marmote SDR: Experimental platform for low-power wireless protocol stack research,” *Journal of Sensor and Actuator Networks*, vol. 2, no. 3, pp. 631–652, 2013.