Smart Wireless Sensor Network Management Based on Software-Defined Networking

Alejandro De Gante
Department of Electrical Engineering and Computer Science
CINVESTAV Unidad Guadalajara
Zapopan, México
sdegante@gdl.cinvestav.mx

Mohamed Aslan[1], Ashraf Matrawy[2]
[1]Department of Systems and Computer Engineering
[2]Carleton School of Information Technology
Ottawa, Canada
mohamedaslan@cmail.carleton.ca
ashraf.matrawy@carleton.ca

***Abstract*—In this position paper, we propose the use of software-defined networking (SDN) in wireless sensor networks (WSNs) for smart management. We argue that smart management using SDN promises a solution to some of inherent problems in WSN management. Furthermore, we propose a generic architecture for a base station in a software-defined wireless sensor network. We also propose a general framework for a software-defined wireless sensor network where the controller is implemented at the base station. We then raise some important questions that need to be investigated in future research in software-defined wireless sensor networks.**

Keywords—Wireless Sensor Network; Software-Defined Networking; Network Smart Management; data-plane; control-plane

I. Introduction

A Wireless Sensor Network (WSN) consists of several wireless devices (nodes) able to collect information from the scenario such as temperature, pressure, sound, light, motions and so on. The wireless connection allows the creation of Ad hoc networks without pre-established physical infrastructure or central management. A WSN can be used for intelligent spaces, habitat monitoring and tracking, among others. Generally speaking, knowing network nodes' position makes it possible to add functionality (e.g. sensing of specifics areas) or improve efficiency (e.g. management) in the network. Node localization is then a key factor for many WSNs applications.

The main weakness of wireless sensor networks is related to the sensor resource limitations, such as processing, memory, energy, and communication capabilities. This weakness may be addressed through smart management of network resources, but this is very hard because in Ad hoc networks each node participates in a decentralized routing to forward data to other nodes, so the determination of which nodes forward data is made independently and dynamically by each node on the basis of network connectivity. This makes global network optimization and smart management a very difficult task.

Software-Defined Networking (SDN) was developed to facilitate innovation and enable simple programmatic control of the network data-path. The separation of the forwarding hardware from the control logic allows easier deployment of new protocols, applications, network visualization, and management [1]. SDN allows network administrators to manage network services through abstraction of lower level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (control plane) from the underlying systems that forwards traffic to the selected destination (data plane) [2].

Network intelligence is centralized in software-based SDN controllers, which maintain a global view of the network. Leveraging the SDN controller's centralized intelligence, it is possible to alter network behavior and deploy new applications and network services in real-time.

This position paper argues for a novel base station architecture for WSN based on the SDN paradigm. We argue that the global view that the SDN controller possesses will help overcome the inherent weaknesses of WSN through smart management. This paper is a first step which includes the description of architectural components as well as some discussion to address the challenges on the use of SDN paradigm in WSNs.

The rest of this paper is organized as follows. In Section II, we present the benefits of SDN approach in WSNs. In Section III, we present a software-defined WSN framework. In Section IV, we describe the proposed base station architecture for the controller. In Section V, we present additional discussions about applying SDN in WSNs. Finally, in Section VI, we present the conclusions and discuss future work.

II. Benefits of Software Defined Networking in Wireless Sensor Networks

SDN is a network paradigm based on the separation of control and forwarding planes in wired networks. Interconnection devices make forwarding decisions only based on a rule set defined by external entities called controllers. The implementation of SDN in wired networks has been extensively discussed in recent work [3, 4, 5]. In wireless sensor networks, SDN can reach a higher potential, as it provides functions that can allow a better collaboration between the base station and forwarding nodes [6, 7, 8, 9, 10]. This section presents how SDN can be useful for WSNs implementations. The main issues are presented below:

A. *Energy Saving*

WSN nodes are energy-constrained devices, creating a limitation on the design and implementation of network protocols and sensor applications. This limitation necessitates only employing energy-efficient protocols or applications in sensor networks. We suggest that SDN could promise an energy-efficient way for sensor network management. Decoupling the control logic from the forwarding one and having a centralized controller that maintains a complete or semi-complete global view of the network can reduce the power consumed by different sensor nodes in order to maintain that view locally. For example, the controller can determine the best routing decisions for an application, and inject these decisions in the form of flow rules into nodes' flow tables, relieving nodes from application-specific routing overhead and reducing the energy consumption. Additionally, SDN can provide a means by which traffic engineering, network slicing and QoS could be efficiently achieved with a lower energy overhead since these tasks will be done at the controller rather than at the energy-limited sensors. It is assumed that the controllers will be running at special base stations that have more power than the average sensor.

B. *Sensor Node Mobility*

In case of mobile sensors, depending on the degree of mobility, the structure of the network frequently changes. Subsequently, the convergence time of the ad-hoc routing protocols varies as well. For example, common ad-hoc routing protocols e.g. DSDV [11] and AODV [12], are distance-vector-based protocols. In case that the network topology changes, distance vector routing protocols require time to converge. If mobility is to be managed by a logically centralized SDN controller, routing decisions and policies could be injected or modified at the sensor nodes on the fly, resulting in a lower convergence time. For example, nodes may update the controller with their mobility information, or the controller may employ a mobility management protocol. Then, the controller can keep track of nodes' locations as they move and update the flow tables of the nodes reflecting new routing decisions.

C. *Network Management*

Network management is a complex and a challenging process including network provisioning, configuration, and maintenance. Traditionally, network management requires the administration of low-level vendor-specific configurations. The use of multi-vendor network components can result in a more complicated management process. Kim and Feamster [13] identified some of the network management problems that SDN promises a neat solution for them.

For sensor networks, the process is even more complicated. A typical sensor node consists of a micro-controller, a number of sensors, a wireless communication chip, and a battery. For example, assume that a new network application or protocol is to be installed on a traditional WSN, the cost of managing the network is relatively high, as re-configuring or maintaining a sensor node usually requires re-programming of the node's micro-controller and altering its ROM. This process sometimes obligates the physical access to the node itself which may not be feasible.

One of the most important and original motives behind SDN was to facilitate the network management process. Basically, SDN transforms the network administration problem to a network programming one. We suggest that the complexity of sensor network management can be dramatically eased with SDN. With the increased flexibility gained by introducing SDN to sensor networks, new routing protocols could be employed on-the-fly without the need of re-programming or re-configuring any sensor nodes. Finally, SDN mitigates the need of compiling different versions of the same "network-applications" for different types of sensor nodes.

D. *Localization Accuracy and Topology Discovery*

Localization is considered a necessity in many sensor networks. In these applications, sensor data without corresponding location information could be considered useless. Given the energy-constrained nature of sensor nodes, we argue that highly-accurate location information could be achieved by employing a centralized localization algorithm [14, 15]. The gathered location information can be aggregated at the centralized controller when SDN is used, where it can be used by a network topology discovery algorithm to further enhance the routing decisions made by the controller. Moreover, the location information can be presented by the controller to the network middleware for the use by the sensor applications.

III. SOFTWARE- DEFINED WIRELESS SENSOR NETWORK FRAMEWORK

In this section, we propose a framework for a Software-Defined Sensor Network. A typical wireless sensor network deployment should include a base station and a number of sensor nodes. The number of sensor nodes and their capabilities varies according to their application as well as their cost. For instance, some sensor networks contain different tiers of sensor nodes. A tier could consist of ordinary sensor nodes, while another could consist of sensor nodes of higher-capabilities.

In the software-defined sensor network framework shown in Figure 1, the sensor nodes do not have to make routing decisions. Instead, they forward or drop packets according to a set of pre-installed rules stored in a special data-structure maintained at every sensor node known as the flow table. In other words, the routes which are considered the best according to application-specific criteria are calculated by the controller (in the base station). In this framework, the controller makes use of location information gathered by any localization technique when finding the best routes. The controller then transforms these decisions to a set of rules that are to be inserted into the nodes' flow tables.

A software-defined sensor network relies on a logically-centralized controller. From the network point-of-view, the controller does not necessarily need be a standalone node. We propose that the control-logic be implemented as a part of the base station.

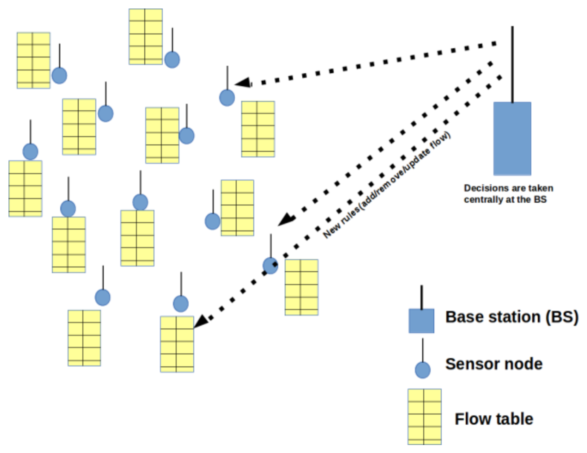
New rules(add/remove/update flow)
Decisions are taken centrally at the BS
Base station (BS)
Sensor node
Flow table

Fig. 1. Software-Defined Wireless Sensor Network Framework

In section IV, we describe the proposed architecture for a base station in a software-defined wireless sensor network.

IV. Base Station Architecture for the Controller

We propose a generic *Controller* node (base station) architecture for wireless sensor networks. This architecture was created in order to address several issues on the management area (mobility, localization and so on.). In addition, this architecture simply provides reconfiguration abilities by the SDN paradigm. Simple network reconfiguration improves management features, such as energy saving, topology discover and so on. Figure 2 shows the proposed architecture emphasizing the units involved in the network management. We will focus on the top two layers (*Middleware* and *Application*).

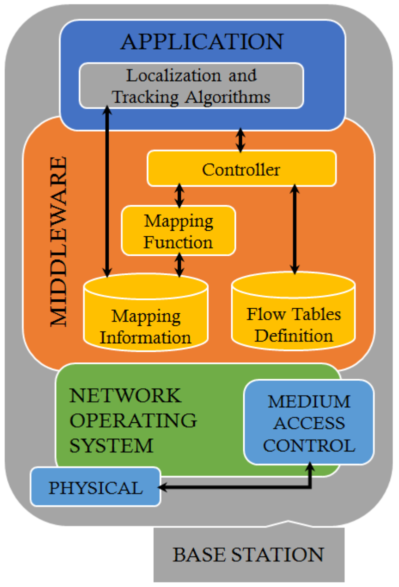

Fig. 2. Base Station Architecture for a Wireless Sensor Network based on Software-Defined Networking. The *Controller* node maintains its functionality throughout 5 layers: (1) Physical; (2) Medium Access Control; (3) Network Operating System; (4) Middleware; and (5) Application

A. The Middleware Layer

The *Middleware layer* was designed to provide a generic control element for the base station. This should also provide a simple interface through which it is possible to define a *flow table* from the *Application*. The main components of this layer are presented below:

- **Controller**

This component is responsible for network control and network management. Since network intelligence is centralized, the *Controller* needs to know an overview of the network state and topology, and for this purpose it uses monitoring messages (WSN monitoring). A monitoring message may contain several types of requests: asking about energy level, distance to the base station (number of hops), neighbor list and status with each one (link quality, time from last connection, response time) and so on. Monitoring messages can be sent using a simple bytes in the payload to represent each request type. The *Controller* may also use messages for routing purposes (control messages).

The *Controller* is supported by a *Mapping Function*, which is responsible for processing the monitoring data received from sensors and make a network map. More details will be discussed below.

Once an overview of the network topology is obtained, the *Controller* can define the *flow tables* based on different aspects of the *Mapping Information*, such as distance, reliability, energy consumption, response time, link quality, etc. It sends the forwarding information related to each node to set the behavior. Moreover, these *Flow Tables Definition* are stored in the *Controller* node, because it must be able to answer any request from the sensors in the network. Note that every node maintains its own *flow table* locally, this is in order to reduce control traffic.

This control logic allows easier deployment of new functionalities and protocols, straightforward network visualization and management, and consolidation of various middle boxes into software control.

- **Flow Tables Definition**

This component is a database, which stores the *flow tables* defined through the *Controller* for every node. These tables can be updated by the *Controller* (management purposes) or by the *Application* through the *Controller* (specific purposes).

TABLE I. Example of Flow Table

nodeX		
Rule	***Action***	***Statistics***
Source = nodeY	Drop	m
Value > x	Forward	n
. . .	. . .	. . .
Destination = nodeZ	Forward	o

Table I shows an example of a *flow table*. This table is composed of one or more sets of *Rule*, *Action* and *Statistics*. A *rule* is a description of the characteristics which are featured by packets belonging to a flow and that must be treated by the node in the same way. Each *Flow Table Set* specifies the *action* which should be executed on all packets satisfying the above *rule*. *Statistics* specifies the number of received packets which have satisfied the rule.

Arriving packets are classified into control packet and data packet. If it is a control packet, it is sent to the *Controller*. If the packet is a data packet, each node checks whether the packet matches one of the rules specified in its *flow table*.

- **Mapping Function**

This component is responsible for processing the monitoring data received in the *Controller*. It has two main duties: one with the *Mapping Information* component and one with the *Controller*.

In the first, the *Mapping Function* takes every neighbor table sent by the sensor nodes and builds a network interconnection map (adjacency matrix). It is also able to process and update information about a network subset (requested via the *Controller*). The *Mapping Function* can be directly called by the *Controller* request (monitoring update) or by *Application* request (update request) via the *Controller*.

In the second, the *Mapping Function* uses the processed information (*Mapping Information*) to provide the *Controller* with general network information such as topology, load, node availability, last monitoring update and statistics. Moreover, when the *Controller* requests a connection path between two nodes, it must be able to provide multiple paths related to distance, reliability, energy level and response time. Each path can be related to one or multiple features to allow the *Controller* to improve resource optimization, load balancing and/or network efficiency. This component is necessary to support the *Controller* operation as it provides routing information.

- **Mapping Information**

This component is a database, which stores the information about network interconnection (*Mapping Information*), such as signal strength, link quality, reliability, and so on.

Mapping information can be accessed and updated through the *Mapping Function* (by *Controller* request) or through *Localization and Tracking Algorithms* (by *Application* request).

The mapping information is organized in an adjacency matrix with size n^2, where n is the number of nodes in the network. If a *nodeX* and a *nodeY* are neighbors (one-hop connection), then the cells [x, y] (collected by *nodeX*) and [y, x] (collected by *nodeY*) will contain the connection information between these two nodes. Otherwise, both cells will be empty. Table II shows an example of connection information between *nodeX* and *nodeY* collected by *nodeX*.

TABLE II. EXAMPLE OF CONNECTION INFORMATION FROM NODEX TO NODEY

nodeX → nodeY	
Datum	***Value***
Signal Strengh (%)	91
Link Quality (%)	73
Reliability (1 – 10)	8
Energy Level (%)	85
Response Time (ms)	42
Last Update (ms)	230

B. The Application Layer

The *Application layer* was designed to provide specific functionality (e.g. temperature monitoring) on the WSN (via base station). This is provided with an interface set to ensure proper interaction with the middleware, mainly with the *Controller* and *Mapping Information* components.

The main function of a sensor network is precisely sensing the environment (temperature, motion, lighting, pressure, and so on) in which it was deployed, process this information and, either use the information or send it to another network. Given the above, the *Application* needs to know the sensed location in order to understand and use the information.

This layer has a location component called *Localization and Tracking Algorithms* (LTA) to address node localization issue and is presented below. This is also provided with a simple interface through which it is possible to access and update mapping information (by LTA).

- **Localization and Tracking Algorithms**

This component is responsible for maintaining accurate information about node position. It has two main duties: one with the *Mapping Information* component and one with the *Application*.

In the first, LTA processes *Mapping Information* to provide the *Application* with estimated node position information. As the mapping information is centralized, then LTA can be based on centralized localization algorithms, usually thought to be more accurate than distributed algorithms [14, 15]. An example of these centralized algorithms is MDS-MAP [16].

In the second, LTA creates a node position array within the *Mapping Information*. This position information can be used for *Mapping Function* in order to update reliability field on Mapping Information and provide a more accurate network view to the *Controller*.

V. DISCUSSION

Assume for a certain application, a WSN needs to be deployed to cover a certain area where it needs to maintain a continuous monitoring. Using the proposed architecture, system behavior could be changed in an easier way. This could be done by modifying the forwarding rules (*Flow Tables Definition*) in the base station (controller) and so, new action policies will be sent to the network devices. Furthermore, maintaining system scalability becomes a simpler task. To extend the network to another area, new devices could be linked to the same WSN (same base station) which should allow for network policies to be sent to the new devices.

In terms of the protocol used for communication, we have to mention OpenFlow. In wired software-defined networks, the OpenFlow [17] protocol can be thought of as the de facto standard that controllers use to communicate with network nodes. However, it is not yet known if OpenFlow in its current specification suits sensor networks or not. Moreover, it is not clear if OpenFlow should be extended or totally replaced by another protocol to better fit-in with the nature and constraints of wireless sensor networks. This is a question that requires further research and investigation.

It is not clear yet, as well, if the use of physically distributed controllers can preserve more energy or outperform a centralized controller, as acquiring a global-view of the network would then require synchronization between a number of distributed controllers.

VI. Conclusion and Future Work

In this paper, we explored the possibility of introducing the concepts of software-defined networks into wireless sensor networks. We argued that the introduction of SDN into WSNs could help in tackling some of the difficult problems in WSNs such as energy saving and network management. We also proposed an architecture for a software-defined wireless sensor networks, where the controller is integrated at the base station.

Among others, the following arguments and questions need to be investigated and answered. First, it is still need to be seen whether to employ a distributed controller or not. How would a distributed controller affect the performance and energy consumption of the sensor network? What is the correct way to synchronize the network global-view at each controller? Second, would the OpenFlow protocol fulfill the needs in such a constrained network? For future work, it is important to try to find answers to these questions. Moreover, the use of a centralized controller raises some security questions that appear with the introduction of software-defined networks in general such as: What would be the effect if attacks are directed towards the controller? What would be the effect of a malicious controller on the network?

Finally, the proposed work and ideas in this paper are simply still at the preliminary proposal stage and further research and investigation are needed to check their impact on WSN in terms of performance, reliability, security and other aspects. In other words, it is yet still to be seen how much of the benefits anticipated in this proposed work and ideas can be actually achieved as well as what limitations or problems they will create in WSN. Moreover, testing in realistic scenarios should create a better understanding of the benefits, limitations and problems in the proposed work and ideas.

Acknowledgement

The first author acknowledges the CONACyT Mexico and the Emerging Leaders in the Americas Program (ELAP) for supporting his academic visit to Carleton University (November 2013 - April2014). The second and third authors acknowledge support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] B. Nunes, A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, Thierry Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys and Tutorials (Under Review). http://hal.inria.fr/docs/00/93/29/82/PDF/hal_final.pdf.

[2] Software-Defined Networking: The New Norm for Networks. White paper. Open Networking Foundation. April 13, 2012.

[3] R. Bennesby, P. Fonseca, E. Mota, A. Passito. An inter-AS routing component for software-defined networks. In Network Operations and Management Symposium (NOMS), 2012 IEEE, pages 138–145. 2012.

[4] A. Bianco, R. Birke, L. Giraudo, and M. Palacin. Openflow switching: Data plane performance. In Communications (ICC), 2010 IEEE International Conference on, pages 1–5, May.

[5] R. Bifulco, R. Canonico, M. Brunner, P. Hasselmeyer, F. Mir. A practical experience in designing an openflow controller. In Software Defined Networking (EWSDN), 2012 European Workshop on, pages 61–66, Oct.

[6] A. Mahmud, R. Rahmani. Exploitation of OpenFlow in wireless sensor networks. Computer Science and Network Technology (ICCSNT), 2011 International Conference on , vol.1, no., pp.594,600, 24-26 Dec. 2011.

[7] T. Luo, H. Tan and T.Q.S Quek. Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks. Communications Letters, IEEE , vol.16, no.11, pp.1896,1899, November 2012.

[8] S. Costanzo, L. Galluccio, G. Morabito, S. Palazzo. Software Defined Wireless Networks: Unbridling SDNs. Software Defined Networking (EWSDN), 2012 European Workshop on , vol., no., pp.1,6, 25-26 Oct. 2012.

[9] P. Dely, A. Kassler, N. Bayer. Openflow for wireless mesh networks. In Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), pages 1–6. IEEE, 2011.

[10] L. E. Li, Z. M. Mao, J. Rexford. CellSDN: Software-Defined Cellular Networks. TR-922-12, May 2012. Department of Computer Science. Princeton University.

[11] S. Das, E. Belding, C. Perkins. Ad hoc on-demand distance vector (AODV) routing. 2003.

[12] C. Perkins, P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In Proceedings of the conference on Communications architectures, protocols and applications (SIGCOMM '94). ACM, New York, NY, USA, 234-244. 1994.

[13] K. Hyojoon, N. Feamster. Improving network management with software defined networking. Communications Magazine, IEEE, vol.51, no.2, pp.114,119. February 2013.

[14] J. Bachrach, C. Taylor. Localization in Sensor Networks. Massachesetts Institute of Technology.

[15] G. Mao, B. Fidan, B. Anderson. Wireless Sensor Network Localization Techniques. Elsevier B. V. 2006.

[16] Y. Shang, W. Ruml. Improved MDS-based localization. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol.4, no., pp.2640,2651 vol.4, 7-11. March 2004.

[17] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 2 (March 2008), 69-74.