Distributed DoS Attack Detection in SDN: Tradeoffs in Resource Constrained Wireless Networks

Gustavo A. Nunez Segura*, Arsenia Chorti‡, and Cintia Borges Margi*
*Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil
‡ETIS UMR8051, CY Université, ENSEA, CNRS, F-95000, Cergy, France
Email: {gustavoalonso.nunez, cintia}@usp.br, arsenia.chorti@ensea.fr

2021 IEEE Statistical Signal Processing Workshop (SSP) | 978-1-7281-5767-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSP49050.2021.9513821

***Abstract*—The Software-defined networking (SDN) paradigm centralizes control decisions to improve programmability and simplify network management. However, this centralization turns the network vulnerable to denial of service (DoS) attacks, and in the case of resource constrained networks, the vulnerabilities escalate. The main shortcoming in current security solutions is the tradeoff between detection rate and complexity. In this work, we propose a DoS attack detection algorithm for SDN resource constrained networks, based on recent results on non-parametric real-time change point detection, and lightweight enough to run on individual resource constrained devices. Our experiment results show detection rates and attacker identification probabilities equal or over 0.93.**

***Index Terms*—Software-defined networking, intrusion detection, wireless sensor networks**

I. Introduction

Software-defined networking (SDN) is a paradigm that centralizes network control decisions and enables the network to be intelligently and centrally programmed. These characteristics simplify network management and provide tools for infrastructure sharing [1].

SDN centralization provides advantages and disadvantages in terms of network security: on one side, the controller's global view has been used to develop new security strategies [2], on the other side, the controller is a single point of failure, which turns SDN-based networks prone to DoS attacks [3] [4]. In the case of resource constrained networks, as wireless sensor networks for Internet of things, SDN vulnerabilities are critical since there are less resources to detect and mitigate attacks. Consequently, current standard SDN security solutions adaptation is not trivial.

Since SDN centralizes the control logic of the network, most of works in the literature propose centralized security solutions. This has benefits, such as a global view of the network and high processing power, but it also requires a constant communication between the network devices and the controller. This means more energy and communication resources consumption. To address this issue, we propose a lightweight DoS attack detection algorithm using change point (CP) analysis to detect anomalies in the network behavior. We execute our proposal using a distributed approach, running the detection algorithm on individual resource constrained nodes, avoiding the packets overhead caused by the centralization.

We simulate grid topologies of 100 nodes, where 10% of nodes are attackers. Our main results show that individual nodes can detect a DoS attack and identify the attacker itself with a probability equal or over 0.93, when being close to the attacker. In addition, we investigate trade-offs between a fully decentralized and a hybrid approach.

II. Related work

Machine learning is a popular approach used for security in SDN since the controller has access to traffic information that could be used to train the algorithms. Bhunia and Gurusamy [5], Ravi and Shalinie [6], and Jia *et al.* [7] proposals have in common that all of them obtained high detection rate results, i.e., higher than 90%, using machine learning techniques. On the other hand, none of these three proposals considered resource constraints. The main reason is because these are OpenFlow-based or require high traffic of packets to monitor the network.

Some proposals focused on resource limitations. Yin *et al.* [8], Miranda *et al.* [9], and Wang *et al.* [10] proposed more lightweight security solutions, but at the cost of detection rates below 90%. However, these works proposed multiple types of attack detection and attacker identification algorithms.

Commonly, security in SDN is centralized, but nevertheless, there are distributed-based proposals in the literature. One reason to use distributed approaches is to avoid control overhead that could saturate vital control links. To address this shortcoming, Fawcett *et al.* [11] proposed Tennison, a framework for scalable network security based on multi-level flow monitoring. Distributed approaches have been used also to detect anomalies in local sub-networks [12]

The main shortcoming in the state of the art is the tradeoff between detection rate and solution complexity. The proposals that attained high detection rate were not suited

for resource constrained networks, and proposals that considered resource limitations did not attain high detection rates. We propose a DoS attack detection algorithm for SDN resource constrained networks, lightweight enough to run on individual resource constrained devices. Results show a detection probability comparable to centralized proposals, but reducing packets traffic, a key shortcoming in centralized solutions.

III. SDN Security Vulnerabilities

In SDNs, the attackers can reach the control plane directly through the controller(s) or through network devices. Control packets flooding attacks are common since these packets have to reach the controller to be processed, which can lead to processing and communication resources exhaustion. The attackers are also able to mislead other network devices and induce them to flood the network.

The SDN controller needs topology information to operate. To this end, the network devices send neighborhood information to the controller for configuration and control decisions. In the case of wireless SDN networks, attackers may hear this information and use it to mislead the controller to take wrong routing decisions. In the case of SDN resource constrained networks, these attacks target specific characteristics. Attackers can launch control plane attacks to saturate flow tables and buffers of devices with low storage capacity. A saturated node may not have space to forward new packets or receive new routing rules. This will trigger a series of packets retransmissions, which means more energy, processing and communication resources consumption. Since the network operation depends on the controller, if this do not take actions, the network devices can exhaust all their resources.

In a previous work [13], we analyzed the impact of the false data flow forwarding (FDFF) attack in SDWSNs. The FDFF attack targets the controller via network devices. First, the attacker sends data packets with unknown flow identifiers to its neighbors. The neighbors receive the packet and check the flow table to determine the action required, without success, thus they ask a rule to the controller by sending a flow rule request packet. The controller receives this packet, calculates the rule and replies sending a flow setup packet. This attack increases the control and processing overhead, and saturate the flow tables on the nodes close to the attackers.

IV. Distributed DoS Attack Detection

From [14] and [15] we know that our CP detection algorithm, based on [16] [17], is able to detect FDFF attacks with a probability over 0.96, and identify the type of attack with a probability exceeding 0.89. In this work, we go further and execute the CP detector running on individual nodes. Our objectives are: first, to evaluate the performance in resource constrained devices, and second, study the tradeoff when running the detectors on every node in the network and running it in clusters.

A. Change point detection

The problem formulation exploits recent results [16], [17] on non-parametric real-time CP detection. We adapted the hybrid offline-online proposal to an entirely online detector [18]. The proposed procedure is applied under the assumption that the observations $\{X_n : n \in \mathbb{Z}\}$ satisfy the generalized dependence concept of L-2 near epoch dependence [19].

To outline the online CP algorithm, let $\{X_n : n \in \mathbb{N}\}$ be the time series of the metric monitored. Using Wold's theorem we can assume that, for $X_1, ..., X_N$, each sample is expressed as $X_n = \mu_n + Y_n$, where $\{\mu_n, n \in \mathbb{N}\}$ is the mean of the time series and $\{Y_n : n \in \mathbb{N}\}$ is a random zero mean term, so that we can rewrite X_n as:

$$X_n = \begin{cases} \mu + Y_n, & n = 1, \ldots, m + k^* - 1 \\ \mu + Y_n + I, & n = m + k^*, \ldots \end{cases} \quad (1)$$

where μ, $I \in \mathbb{R}^r$, represent the mean parameters before and after the unknown time of possible change $k^* \in \mathbb{N}^*$ respectively. The term m denotes the length of an initial period assuming no change on the mean value, i.e, $\mu_1 = \cdots = \mu_m$. During this period, our detector "learns" in real-time the statistics of the observed time series, and, the mean value in particular. Finally, the statistical hypothesis test is articulated as: $H_0 : I = 0$, $H_1 : I \neq 0$.

The online analysis is a stopping time stochastic process defined as:

$$\tau(m) = \begin{cases} \min\{l \in \mathbb{N} : TS_{on}(m,l) \geqslant F(m,l)\}, \\ \infty, \text{ if } TS_{on}(m,l) < F(m,l) \ \forall l \in \mathbb{N}, \end{cases} \quad (2)$$

where $TS_{on}(m,l)$ is the detector, calculated online for every l, and $F(m,l)$ is the given threshold; with properties $\lim_{m\to\infty} Pr\{\tau(m) < \infty | H_0\} = \alpha$, ensuring that the probability of false alarm is asymptotically bounded by $\alpha \in (0,1)$, and, $\lim_{m\to\infty} Pr\{\tau(m) < \infty | H_1\} = 1$, ensuring that under H_1 the asymptotic power is unity. Under these conditions, $F(m,l) = cv_{on,\alpha} g(m,l)$, where the critical value $cv_{on,\alpha}$ is determined from the asymptotic distribution of the detector under H_0 and the asymptotic behavior achieved by letting $m \to \infty$. The weight function is defined as,

$$g(m,l) = \sqrt{m}\left(1 + \frac{l}{m}\right)\left(\frac{l}{l+m}\right)^{\gamma} \quad (3)$$

where the sensitivity parameter $\gamma \in [0, 1/2)$.

The online algorithm uses the standard CUSUM detector [20], with test statistic denoted by TS^{ct}_{on}. Its corresponding critical value is denoted by $cv^{ct}_{on,\alpha}$ and the stopping rule by $\tau_{ct}(m)$. The sequential CUSUM detector is denoted by $E(m,l) = \left(\overline{X}_{m+1,m+l} - \overline{X}_{1,m}\right)$.

The standard CUSUM test is expressed as:

$$TS^{ct}_{on}(m,l) = l\widehat{\Omega}_m^{-\frac{1}{2}} E(m,l), \quad (4)$$

978-1-7281-5767-2/21/$31.00 ©2021 IEEE

where $\widehat{\Omega}_m$ is the estimated long-run covariance, defined as in (4), that captures the dependence between observations. Then, the stopping rule $\tau_{ct}(m)$, is defined as:

$$\tau_{ct}(m) = \min\{l \in \mathbb{N} : \|TS^{ct}_{on}(m,l)\|_1 \geq cv^{ct}_{on,\alpha} g(m,l)\}, \tag{5}$$

where the ℓ_1 norm is involved to modify TS^{ct}_{on} so that it can be compared to a one dimensional threshold function. The critical value, $cv^{ct}_{on,\alpha}$, is derived from the asymptotic behavior of the stopping rule under H_0:

$$\lim_{m\to\infty} Pr\{\tau(m) < \infty\} \tag{6}$$

$$= \lim_{m\to\infty} Pr\left\{ \sup_{1\leqslant l\leqslant\infty} \frac{\|TS^{ct}_{on}(m,l)\|_1}{g(m,l)} > cv^{ct}_{on,\alpha} \right\} = \alpha \tag{7}$$

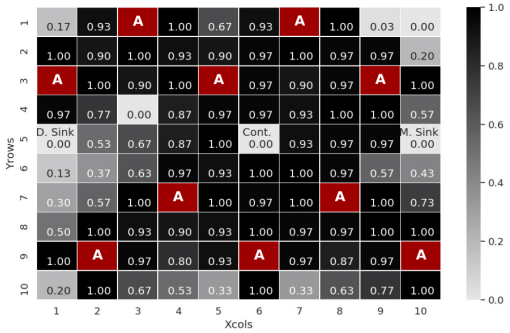
B. DoS attack detection: implementation

In our distributed proposal, every node is able to monitor a time series using its own metrics and execute the CP detector algorithm, and also, each node is able to send metrics samples to a cluster head (CH). In the second case, the CH is in charge of constructing the time series of the cluster and execute the detection algorithm. Such approaches, in which hardware behavior is monitored to identify anomalies, hints to further integration with other approaches for the domain of physical layer security [21] and introducing security controls at all layers.

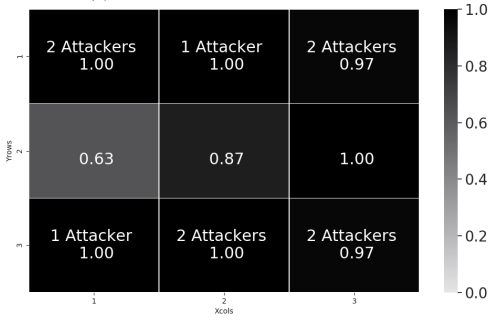
A security application was programmed in every node. This application manages the sampling and the detection algorithm execution. The algorithm initiates constructing a time series of 200 samples ($m = 200$), from where it extracts the statistical information that will use during the online CP detection. When the online phase starts, the algorithm continues storing samples. In the case no CP is detected during the first 50 samples ($l = 50$), these samples are added to the 200 samples taken before to extract new statistical information. But, if a CP is detected, the application rises an alarm and informs the controller about the situation.

Since the FDFF attack increases the message exchange activity, we decided to monitor the transmitting time on every node, i.e, the number of ticks the radio module remained turned on transmitting packets. From previous experiments simulating a real monitoring application [15], we learned that $\gamma = 0$ and $m = 200$ maximize the detection rate. In this work we used these values as well.

The security module was implemented in C language using Contiki-3.0 [22], an operating system for WSN and IoT, and IT-SDN [23], an SDWSN framework developed by our research group. For transmitting time sampling we used Energest [24], a power management module for Contiki. For control packets sampling we implemented a counter for this specific type of packets.



(a) FDFF detection: individual nodes



(b) FDFF detection: clusters

Fig. 1: Distributed FDFF attack detection

V. Results and Analysis

Our analysis follows two approaches: detection performance and implementation overhead. For detection performance we analyzed the attack detection probability and attacker identification probability. For implementation, we analyzed the packets overhead and memory usage. Both scenarios were simulated on Cooja [25], simulating grid topologies of 100 nodes, where 10% of nodes are attackers and emulating sky motes.

A. Detection performance

Fig. 1 shows the detection probability heatmap for both the distributed and hybrid approaches. For the case where each individual node is running the detector (Fig. 1a) the attackers' position is represented with an "A". For the case where the detector is running on clusters (Fig. 1b), the heatmap shows also the number of attackers in each cluster.

In Fig. 1a we observed that the detection probability is higher around the attackers and around the controller, with values between 0.93 and 1.00. We expected this since the FDFF attack targets the control plane through the attackers' neighbors, thus the attack has direct impact on the transmitting time mean value of these nodes. The CPs detected on the nodes at two or more hops from the attackers reflect the impact of this attack in the whole network. The increase in the transmitting time of these nodes is caused mainly by the increase in the control packets forwarding. However, individual nodes are not able to determine if the anomaly is caused by an attack or

978-1-7281-5767-2/21/$31.00 ©2021 IEEE

a normal behavior. As future work, we intend to address this in a centralized application that analyzes the alarms detected to determine if in fact the network is under attack.

The detection probability results for the clustering case in Fig. 1b showed that in the clusters where there are one or two attackers, the detection probability was equal or above 0.97. In the clusters without attackers, the detection probability was between 0.63 and 1.00. Similar to the case running the detector on individual nodes, high detection probability results in clusters without attackers mean that this attack impacts on the whole network.

The detection probability results provided two important insights: (i) we were able to detect an FDFF attack when monitoring the transmitting time in either individual nodes or clusters, and (ii) the detection probability was lower on the nodes at two or more hops from the attackers. Based on (ii), we implemented an attacker identification algorithm for the case running on individual nodes. First, every time a node receives a data packet with an unknown flow, it saves the identification address of the sender in a vector for suspects. Then, if a node detects a CP on the transmitting time, it sends an alarm to the centralized security module and informs the address of a suspect. To determine the suspect, the node checks the last ten addresses saved in the vector and choose the one with the highest frequency. We chose ten samples because, according to [15], the slower detection when $\gamma = 0$ takes around ten samples. When the security module receives the alarm, sets a counter for every suspect. If the counter is equal to the number of neighbors of the corresponding suspect, the suspect is declared as attacker.

The heatmap in Fig. 2 showed that using this algorithm we were able to correctly identified all the attackers with a probability equal or above 0.93. In addition, the false positives were equal to zero. On the other hand, this was possible only running the detector on every individual node. In the clustering approach, groups without attackers inside also obtained high detection rates, which excluded the possibility of tracking the attacker based only on the alarm received from the cluster.

B. Implementation comparison

In the clustering approach, the cluster head is in charge of constructing the time series for the whole cluster and execute the CP detector. To accomplish this, all nodes have to monitor their transmitting time and send a sample to the cluster head, periodically. The cluster head sums up all the samples received per period and the result is a new sample of the cluster's time series. Conversely, when the CP detector is running on individual nodes, each one constructs its own transmitting time time series and executes the CP detector.

The clustering approach has the benefit that it reduces memory and processing overhead, since the detector is running on the cluster heads only. Conversely, it increases

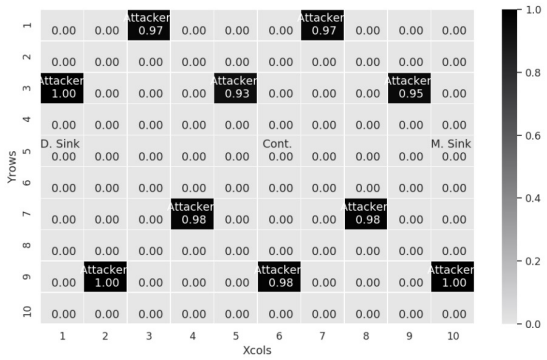


Fig. 2: Attacker identification probability:

the packets traffic, since all nodes have to periodically send a transmitting time sample to the cluster head. In terms of memory, the time series construction and the CP detector implementation, for one metric, represents 5956 B. In our specific case using sky motes, this value represents 12.40% of the total memory.

One security risk when using clustering approaches, is the possibility of the cluster head to be an attacker. One way to solve this is using secure cluster head selection algorithms [26], but this requires more memory, processing, and communication resources, which are already scarce in our case. One benefit of running the detector on every node is that we avoid this risk since the attack detection does not depend on one or few nodes only.

VI. Conclusion

In this work we propose a distributed DoS attack detection for software-defined resource constrained wireless networks, based on CP detection. Our proposal is lightweight enough to run on very limited devices and detect FDFF attacks with a probability between 0.93 and 1.00, comparable to detection results in centralized proposals.

This proposal was evaluated running the detector on every node and running the detector in clusters. Results showed that both approaches obtained high detection rate. Additionally, we were able to identify the attackers without increasing the packets traffic when running the detector on every node. Comparing the implementations of both approaches, the clustering approach requires less memory on every node, while running the detector on every node reduced the packets traffic.

As future work, we envisage to develop a full implementation of distributed and centralized approaches and compare their performance based on security and network performance metrics.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and by the ELIOT project (ANR-18-CE40-0030 / FAPESP 2018/12579-7).

REFERENCES

[1] 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.

[2] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in Software Defined Networks: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, Fourthquarter 2015.

[3] M. Pal Singh and A. Bhandari, "New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges," *Computer Communications*, vol. 154, pp. 509 – 527, 2020.

[4] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 325–346, Firstquarter 2017.

[5] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in IoT using SDN," in *27th Int. Telecommun. Netw. and Appl. Conf. (ITNAC)*, Nov 2017, pp. 1–6.

[6] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of ddos attack in iot via sdn-cloud architecture," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3559–3570, 2020.

[7] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "Flowguard: An intelligent edge defense mechanism against iot ddos attacks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, 2020.

[8] D. Yin, L. Zhang, and K. Yang, "A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework," *IEEE Access*, vol. 6, pp. 24 694–24 705, 2018.

[9] C. Miranda, G. Kaddoum, E. Bou-Harb, S. Garg, and K. Kaur, "A collaborative security framework for software-defined wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2020.

[10] R. Wang, Z. Zhang, Z. Zhang, and Z. Jia, "ETMRM: An Energy-efficient Trust Management and Routing Mechanism for SD-WSNs," *Computer Networks*, vol. 139, pp. 119 – 135, 2018.

[11] L. Fawcett, S. Scott-Hayward, M. Broadbent, A. Wright, and N. Race, "Tennison: A distributed sdn framework for scalable network security," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2805–2818, 2018.

[12] J. Shu, L. Zhou, W. Zhang, X. Du, and M. Guizani, "Collaborative intrusion detection for vanets: A deep learning-based distributed sdn approach," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020.

[13] G. A. N. Segura, C. B. Margi, and A. Chorti, "Understanding the Performance of Software Defined Wireless Sensor Networks Under Denial of Service Attack," *Open Journal of Internet Of Things (OJIOT)*, 2019, special Issue: Proc. Int. Workshop Very Large Internet of Things (VLIoT 2019) in conjunction with the VLDB 2019 Conf. Los Angeles, United States.

[14] N. S. Gustavo, S. Skaperas, A. Chorti, L. Mamatas, and B. M. Cintia, "Denial of Service Attacks Detection in Software-Defined Wireless Sensor Networks," in *SecSDN IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020.

[15] G. A. N. Segura, A. Chorti, and C. B. Margi, "Multimetric online intrusion detection in software-defined wireless sensor networks," in *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, 2020, pp. 1–6.

[16] S. Skaperas, L. Mamatas, and A. Chorti, "Early Video Content Popularity Detection with Change Point Analysis," in *IEEE Global Commun. Conf. (GLOBECOM)*, Abhu-Dhabi, United Arab Emirates, Dec. 2018.

[17] S. Skaperas, L. Mamatas, and A. Chorti, "Real-Time Video Content Popularity Detection Based on Mean Change Point Analysis," *IEEE Access*, vol. 7, pp. 142 246–142 260, 2019.

[18] G. A. N. Segura, A. Chorti, and C. B. Margi, "Centralized and distributed intrusion detection for resource constrained wireless sdn networks," 2021.

[19] J. Davidson, *Stochastic limit theory: An introduction for econometricians*. OUP Oxford, 1994.

[20] S. Fremdt, "Asymptotic distribution of the delay time in page's sequential procedure," *Journal of Statistical Planning and Inference*, vol. 145, pp. 74 – 91, 2014.

[21] A. Chorti, C. Hollanti, J. Belfiore, and Harold Vincent Poor, *Physical layer security: A paradigm shift in data confidentiality*, ser. Lecture Notes in Electrical Engineering. Germany: Springer Verlag, 2016.

[22] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[23] R. C. A. Alves, D. A. G. Oliveira, G. A. Nunez Segura, and C. B. Margi, "The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks," *IEEE Access*, vol. 7, pp. 115 093–115 108, Aug 2019.

[24] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, ser. EmNets '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 28–32. [Online]. Available: https://doi.org/10.1145/1278972.1278979

[25] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, Nov 2006, pp. 641–648.

[26] A. Shankar, N. Jaisankar, M. S. Khan, R. Patan, and B. Balamurugan, "Hybrid model for security-aware cluster head selection in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 9, pp. 68–76(8), April 2019.