Received ; | Revised ; | Accepted
DOI: xxx/xxxx

RESEARCH ARTICLE

Efficient Controller Placement and Re-election Mechanism in Distributed Control System for Software Defined Wireless Sensor Networks

Hlabishi I. Kobo*[1] | Adnan M. Abu-Mafhouz[1,2] | Gerhard P. Hancke[3,1]

[1]Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0084, South Africa
[2]Council for Scientific and Industrial Research, Pretoria 0001, South Africa
[3]Department of Computer Science, City University of Hong Kong, Hong Kong, China

Correspondence
*Hlabishi I. Kobo, Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0084, South Africa. Email: hlabishik@gmail.com

Summary
Software Defined Wireless Sensor Networking is a new wireless networking paradigm formed by applying Software Defined Networking to Wireless Sensor Networks. Fragmentation based distributed control system offers an efficient way of distributing the SDWSN control services across the network. Fragmentation aims to bring the control services closer to the infrastructure network in order to reduce the propagation latency. It also aims to improve the scalability, reliability and performance of the network. The Fragmentation model is earmarked to play a huge role in stimulating participation of SDWSN in IoT. To realise this, we optimise the model for deployment and integration, as well as for operational efficiency. We consider two aspects: controller placement and controller re-election after failure. This paper discusses the controller placement techniques suitable for SDWSN and the controller replacement in a case of failure for SDWSN. The controller placement and the controller replacement mechanism were both evaluated and the results proved to be effective and efficient.

KEYWORDS:
Software Defined Networking, Software Defined Wireless Sensor Networking, Controller Placement, Controller Re-election Mechanism

1 | INTRODUCTION

The advent of Software Defined Networking (SDN) to computing and networking has inspired a great paradigm shift from conventional practises. The adoption of the SDN model has been growing at a rapid pace in recent times both in the industry and academia. The SDN model is currently being adopted and applied to various computing models such as Cloud, Fog, Edge etc., as well as to various networking models such as mobile, wireless, enterprise[1]. SDN is also seen as a catalyst for the imminent Internet of Things (IoT) paradigm. The SDN model bring about innovation, simplicity, and evolution to networking. The SDN model is premised at the decoupling of the control and data forwarding in the network elements such as the switches or the sensor nodes[2,3]. It separates the control functionality and centralise it in a SDN controller. This leaves the network elements as shallow devices with only the data forwarding functionalities. The data forwarding is governed by the controller, which creates forwarding rules known as flow rules.

[0]**Abbreviations:** SDN, Software Defined Networking; SDWSN, Software Defined Wireless Sensor Networking; WSN, Wireless Sensor Networks, IoT, Internet of Things

Software Defined Wireless Networking (SDWSN) is formed by combining the SDN model with Wireless Sensor Networks (WSN). The application of SDN in WSN aims to improve the usability and applicability of the WSN, particularly as it is envisaged to play a prominent role in the IoT[4,5,6,7]. WSNs are inherently resource constraint with limited memory, energy, processing, and data rates[8,9,10]. This decoupling offers a necessary respite as most of the energy intensive functions are moved from the sensor device to the central controller[11].

The centralisation of the controller brings many benefits but also arouses potential challenges of security, reliability, and performance[12,13]. A single central controller is a potential target of malicious attacks. Any attack would render the network dysfunctional because the controller is in charge of the entire network. This also raises issues of reliability because in case the controller fails due to various network dynamics, the network will again cease to operate. On the other hand, the controller could be overwhelmed by all the updates and requests that it receives. This could negate the performance of the network.

As a result of the above challenges, distributed controllers have been punted as a potential resolve. In the traditional enterprise networks, many distributed SDN controllers have bee proposed and developed such as ONOS[14], OpenDaylight[15], Hyperflow[16], Kandoo[17], and Elasticon[18]. In SDWSN, the development of these distributed controllers is still lacking, mainly due to the infancy of the SDWSN. However, there are few solutions such as SDN-WISE[19] and TinySDN[20,21]. Kobo *et al.*[22] proposed and developed a distributed control system called Fragmentation for the SDWSNs, taking into consideration the inherent resource limitations. It fragments the SDWSN into different clusters, with each cluster having its own controller service. Thus it comprises a two level control architecture consisting of local controllers and a global controller. The main aim is to reduce the distance between the sink nodes and the local controllers as well as the distance between the local controllers and the global controller.

The challenge is to ensure that the placing of the local controllers and the global controller(s) follow a procedure that will minimise latency and provide resilience. This would also determine the number of local controllers for any given network. The strive of the minimum distance has to be maintained even in the event of failure. Thus the other challenge is to ensure that the controller replacement after failure takes distance into account. This paper seeks to optimise the Fragmentation model[22] for real world deployment by optimal controller placement and efficient controller re-election mechanism. Thus the purpose of this paper is to ensure that the distributed control system for SDWSN through Fragmentation is simplistic and usable. This will facilitate the integration of SDWSNs with other IoT networks. The main contributions of this paper are:

- To enhance the Fragmentation-based distributed control system for SDWSN.
- To investigate an optimal controller placement method for the distributed control system in SDWSN using the fragmentation model.
- To investigate an efficient controller re-election mechanism for the controller replacement in a case of failure.

The rest of the paper is organised as follows: In Section 2, we provide the background which includes overviews of the Fragmentation model, the controller replacement, and the controller re-election. In Section 3, we discuss the controller placement problem in SDWSN. This section includes the proposed SDWSN controller placement solution, experimental evaluation, and results and discussion. Section 4 discusses the controller re-election mechanism in a case of controller failure. This section also includes the proposed re-election solution, experimental evaluation, and results and discussion. This paper is concluded in section 5.

2 | BACKGROUND

2.1 | Overview of the Fragmentation model

Fragmentation is a way of distributing the control system of the SDWSN proposed by Kobo *et al.*[22]. The aim of this model is to bring controller services closer to the network infrastructure. The reason being that SDWSNs are inherently resource constraint; and thus for optimal efficiency, reliability, and performance, the controllers are distributed across the network. This reduces the distance between the sink nodes and the controllers thereby reducing the propagation latency. Fragmentation comprises a two level architecture consisting of local controllers and a global controller(s). The local controllers are small, lean, and inexpensive. They are deployed in a form of clusters and operates independently of each other. Each local controller has knowledge of only the cluster it is controlling. This ensures that there is a faster response between the sink nodes and the local controllers. It uses consistency data models and algorithms that allows much independence and parallelism amongst the local controllers.

On the other hand the local controllers relays the updates of their respective clusters to the global controller, which has a knowledge of the entire the network. Applications that need a global knowledge are executed at the global level whilst the rest at the local level. Figure 1 below depicts the architecture of the fragmentation model of the distributed control system for SDWSN. This architecture ensures the data under observation receives appropriate controller services in a fast and efficient manner. The local controllers do not have knowledge of the other clusters because the SDWSN carries a sensory data and therefore do not need the global knowledge of the network to operate. Detailed charecteristics of each component can be found in[22].

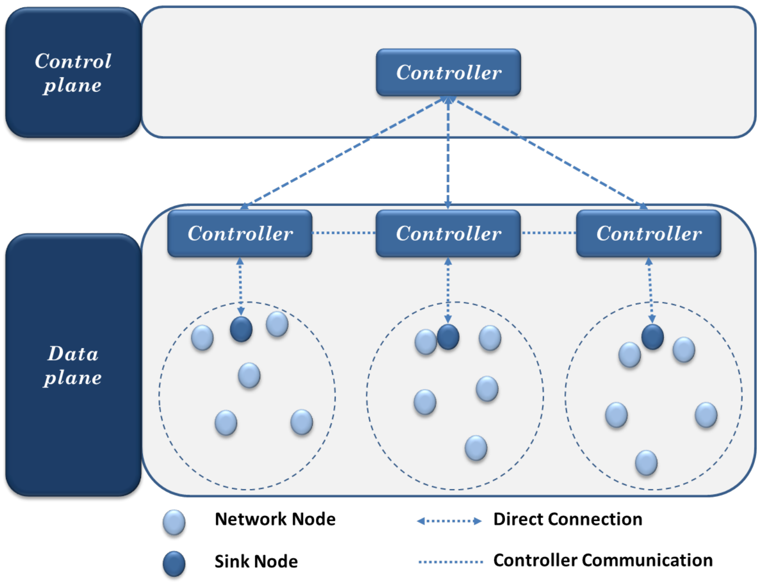


FIGURE 1 The architecture of the Fragmentation model of controller distributions[22].

2.2 | The controller placement problem

The fragmentation model fragments the control logic so that each fragment controls a particular portion of the network. This, among other things, reduce the distance between the sink nodes and the controller and thus ensures that the network latency is minimised. In a large and operational network, the placement of these local controllers should be diligent and systematic; this is referred to as the controller placement problem.

The controller placement problem is a concept chiefly studied in the traditional SDN space. Its fundamental purpose is rooted in minimising latency between the nodes (sensors or switches) and controller(s). This problem is, however, unexplored in the SDWSN paradigm. Its purpose is also to minimise the latency between the sink nodes and the controllers to ensure optimal performance. The overall and dominant conclusion from the existing controller placement work is that there is no recipe[23] nor any placement rules[24] that apply to all networks, but they offer guidelines through which optimal placement can be found. Therefore, there is no single best controller placement solution, especially when several performance and resilient metrics are used. There is only a trade-off between these metrics[23,24]. Heller *et al.*[24] further state that the placement problem has been well explored and no new theoretical insights are to be expected.

This problem has been largely studied in the traditional SDN networks but not in the SDWSN. Heller *et al.*[24] were the first proponents of this problem which revolved around answering the two questions:

1. How many controllers are needed?
2. Where in the topology should they go?

These are two fundamental questions that need to be answered and they apply to SDWSN too. Heller's work presents comprehensive guidelines for operators to deploy multiple controllers effectively in their SDN networks. This research work set the scene for this problem. The work was aimed at determining the number of controllers to deploy and where in the topology they ought to be located. The authors determined that the controller placement problem was an NP-hard problem, at least as hard as the hardest problem. Nondeterministic Polynomial is a set of computational problems that can be verified in a polynomial time by a deterministic Turing Machine. Heller *et al.*[24] optimised the controller placement problem, based on the latency metric and they developed two latency cases. The first is referred to as average latency, which is modelled as:

Given a network graph $G(V, E)$, let the edge weights represent propagation latencies, d(v,s) represent the shortest path from node $v \in V$ to $s \in V$, $n = |V|$ is the number of nodes, S will be the number of controllers to choose from, S' is the number of controllers to be placed such $|S'| = K$, the average latency for a placement of controllers $L_{avg}(S')$ is modelled as:

$$L_{avg}(S') = \frac{1}{n} \sum_{v \in V} \min_{s \in S'} d(v, s) \tag{1}$$

The worst-case latency is modelled as:

$$L_{wc}(S') = \max_{v \in V} \min_{s \in S'} d(v, s) \tag{2}$$

The average latency uses a k-median[25,26] optimisation approach. The k-median is a clustering method which seeks to find k-cluster centres such that the sum of the distances between the centre and all other points in the cluster are minimised. The k-median strives to minimise the 1-norm distances between each point and its centre in the cluster. The k-median was borne as an improvement of the much-studied k-means approach which also minimises the distance between the centre and the cluster points. The main difference is the variables they used; according to the names, k-median uses the median while k-means uses the mean. The reason behind the k-median is that k-means is vulnerable to outliers[25]. The worst-case latency metrics by Heller *et al.*[24] uses the k-centre clustering technique. The k-centre aims to minimise the maximum distance between the centroid and the points in the cluster[27], thus it minimises the n-norm distance. The authors further deduced that one controller is sufficient to fulfil the existing reaction-time requirements although not fault tolerance.

Hock *et al.*[23] extend this work by adding resilience requirements, thereby showing that in instances where a single controller complies with the latency condition, more controllers would be necessary for the resilience requirement. In addition to controller failure, the authors consider intercontroller latency, network disruption and load balancing. Hock's work is formally referred to as Pareto-based Optimal COntroller placement (POCO)[23,28]. The POCO framework is available on opensource. The POCO uses the worst-case latency from Heller *et al.*[24] because they argue that the average does not consider the worst-case scenarios. They further distinguish between a failure-free scenario and a controller failure scenario. A failure-free scenario is the optimisation where the failure of the controller nodes is less, whilst the controller failure scenario occurs when the controller failure is expected to reach k-1, thus all fails but 1. Hock's failure-free latency metric is modelled as in equation 2, for distinction purposes, we shall refer to it as the maximum latency for failure-free optimisation:

$$L_{ff}(S') = \max_{v \in V} \min_{s \in S'} d(v, s) \tag{3}$$

The controller failure optimisation, which is for the worst-case optimisation, is referred herein as maximum latency for controller failure optimisation and modelled as:

$$L_{cf}(S') = \max_{v \in V} \max_{s \in S'} d(v, s) \tag{4}$$

The controller failure effectively doubles the optimisation, and thus ensures that the maximum latency value covers all other deployed controllers k such that any k-1 controller failure is catered for. The first case in equation 3 equally distributes the controllers across the breadth of the network, while the second case in equation 4 moves all controllers towards the centre of the network, ensuring that in a worst-case scenario, the latency remains within an acceptable range. The authors conclude that the first scenario suffers from high latencies in worst-case scenarios but lower latencies in failure-free scenarios whilst the second performs better in the worst-case scenarios and worst in failure-free scenarios. This work also investigated the intercontroller latency and load balancing.

This work is extended in Lange Bari *et al.*[29] by using heuristics methods to cater for large-scale and dynamic networks. Bari *et al.*[30] propose a dynamic provisioning of controllers according to their activeness in the network. Lin *et al.*[31] propose a controller placement algorithm that aims to determine the number of controllers and their location on the network. They also prove that controller placement is an NP-completeness problem. The work of other researchers such as Ishigaki *et al.*[32] and Muqaddas *et al.*[33] focuses on reducing the intercontroller latency instead of the node/switch-to-controller latency.

The closest to SDWSN is the work by Reze *et al.*[34] which deals with the efficient deployment of multi-sink and multi-controllers in WSN for a smart factory. The authors optimise the placement problem as Integer Linear Programming (ILP) which ensures that every sensor node is covered by at least one sink node and at least one controller node. The distance estimation used Dijkstra's shortest-path algorithm.

2.3 | The controller re-election problem

The golden rule in distributed SDN is that a device should be connected to at least one controller but can only be controlled by at most one controller at any time. The role of the consensus algorithms is to maintain this and to ensure that leadership changes are consistent, procedural, and efficient. Paxos[35,36] and Raft[37,38] are two of the most popular consensus algorithms used in SDN-distributed controllers.

These algorithms are normally applied to in-memory data grid frameworks such as ZooKeeper[39], Hazelcast[40] and Atomix[41]. Both Zookeeper and Atomix use Raft as their consensus algorithm, while Hazelcast uses Best Effort and Anti-entropy. The ONOS framework used Hazelcast up to version 1.4, from which they started to use Atomix (because of the split-brain problem). Atomix uses a strong consistency data model, which is not suitable for the fragmentation model[22]. The fragmentation model is based on the eventual consistency data model, thus eventual consistency ONOS over strong consistency ONOS. The Hazelcast framework also allows a choice between strong or eventual consistency.

2.4 | Summary of discussion

The SDN model brings many benefits to networking and computing. It's application to WSNs to form the SDWSN has re-ignited so much interest in the Wireless Sensor Networks especially with the advent of the Internet of Things. There are so many benefits of the SDWSN and equally so, are the challenges. In our previous work[1], we with Kobo *et al.*[1] comprehensively discussed the concept of SDWSN, with much focus on architectures, routing, security, standards, and network management. Several challenges were highlighted, and one such was the centralisation of the control system[12] which led to the development of the fragmentation model[22]. This paper extends the fragmentation model[22] by developing an efficient controller placement and controller re-election mechanisms.

The two problems discussed in this paper are very critical to the development of the SDWSN. Most of the SDN-based solutions focuses on the traditional enterprise networks[1,42]. Thus the current state of the art solutions on controller placement and controller re-election after failure are based on the traditional SDN networks. This paper therefore fills the gap left for the SDWSN.

3 | THE CONTROLLER PLACEMENT IN SDWSN

Most of the research works above are based on the traditional SDN, except Faragardi *et al.*[34]. However, Faragardi *et al.*[34] deals with the placement of the sink nodes and the multi-controller placement, which follows the conventional way of distribution, against which this research work argues. There are several localisation techniques for the traditional WSNs[43], however , this paper focuses on the placing of the controller for the SDWSNs. Overall, the controller placement footprint is lagging behind in SDWSN. This can be attributed to the recency of the SDWSN; most of the work is still in development stages and focuses primarily on the architectural framework. However, SDN solutions provide insightful guidelines for developing SDWSN solutions. The only major distinction is in the varying traits of the networks, with SDWSN typified by limited resources. Therefore, most SDN solutions need to be customised to work in SDWSN. The controller placement problem in SDWSN is vital for optimal performance and efficiency of the network. In traditional SDN, the problem seeks to reduce the latency between the SDN switches and the SDN controllers. However, in SDWSN, particularly our solution, it is between the sink nodes and the controllers. The two-level architecture adds a little latency; hence, the challenge is to ensure that this latency is kept at a very minimum to insignificant levels.

In most situations, the controller placement problem entails controller failure, network disruption, load balancing, and inter-controller latency [2]. The fragmentation model deals with most of these problems. The inter-controller problem is out because there are no data exchanges between the local controllers, except with the global controller, in which case, the latency does not affect the convergence of data. Network interruptions are rife in SDWSNs but do not affect the operation of the control logic, otherwise the latency between the sink nodes and the controller is minimised by bringing the controller service closer.

3.1 | SDWSN controller placement for Fragmentation

The purpose of this exercise is to optimise the fragmentation model with the aim of minimising latency between the sink nodes and the local controllers, as well as between the local controllers and the global controller. Unlike the solutions reviewed above, this focuses on the sink-to-controller latency. We assume that the placement of the sensor nodes and sink nodes is in accordance with the demands of the network or the service being provided. Therefore, the control framework needs to respond to the needs of the network not the other way around.

As stated in Heller *et al.*[24] the controller placement problem has been studied extensively and no new theoretical framework is likely. The theoretical framework as proposed by Heller *et al.*[24] and further by Hock *et al.*[23] provides a fundamental baseline. We apply this in the SDWSN, particularly for the fragmentation model. Heller *et al.*[24] provided two use-case scenarios, the average case which uses k-median and the worst case which uses k-center. These optimisation techniques are applicable to SDWSN. We adopt the average-case scenario by Heller using k-means. Although Heller states that this formula is for k-median, it actually applies to both because it produces the mean and the median, as well as the maximum value used in k-center; it therefore depends on the use and choice of the criterion. We use this to optimise for latency, we deal with resilience in section 4. The method uses a k-means clustering approach which minimises the distance between the node (sink node) and the cluster centre (controller). This will ensure that the latency between all sink nodes and the controller is minimised. This is modelled as in equations 1; the fragmentation latency L_{frag} is modelled as:

$$L_{frag}(S') = L_{avg}(S') = \frac{1}{n}\sum_{v\in V}\min_{s\in S'} d(v,s) \tag{5}$$

The k-means clustering partition data observations of any n-by-p matrix into k-clusters, it returns an n-by-1 vector consisting of the cluster centres called centroids[44]. K-means uses different distance metrices to compute the $d(v,s)$ such as Euclidean (default), Cosine, Cityblock, Hamming, and Correlation. It uses a k-mean++ algorithm for choosing the best k, which is an improvement on the original criterion which often led to poor clustering[44].

The latency threshold in SDWSNs should be much lower than it can be in traditional SDN, owing to the lower data rates of the sink nodes. This will augur well for the fragmentation model, which seeks to bring control logic closer to the nodes. The placement of the global controller is also important, and therefore, we use the second latency optimisation, the worst case as proposed in[23]. This entails minimising the latency between the local controllers and the global controller(s). This effectively doubles the optimisation and locates the global controller at the centre of the entire network. The formula follows that in equation 4 which is modelled as latency for the global controller L_{gc}:

$$L_{gc}(S') = \max_{v\in V}\ \max_{s\in S'} d(v,s) \tag{6}$$

3.2 | Experiment

We use Matlab to model the controller placement. Matlab is rich in tools used for algorithm testing and mathematical modelling. Accordingly, there are built-in tools for clustering. The Matlab built-in k-means uses a k-means++ algorithm. The formula used in Matlab is[44]:

$$[idx, C, sumd, D] = kmeans(X, K, Name, Value) \tag{7}$$

Where:

- X is the matrix under observation i.e. graph G above.
- K is the specified number of clusters required.
- The Name-and-Value pair represents the distance metric and its name i.e. 'Distance', 'cityblock'.
- idx returns a vector which contains the cluster indices of observation matrix X (graph).
- C returns the k-cluster centroid locations, these are the centres of the partitioned clusters.
- sumd is a k-by-1 vector which contains the within-cluster sums of each point to cluster distance in the cluster.
- D returns a vector of n-by-k matrix of all distances from each point to every centroid.

Matlab simulation defines nodes with no specification of the type. Therefore the set of nodes in our experiment represents the sink nodes amongst which we want to place the k number of controllers. We placed fifteen (15) sink nodes at different locations to determine the locations of the three (3) local controllers and the global controller. We used actual latitude and longitude coordinates as listed in table 1 below. Table 2 lists the weights of the coordinates using the Haversine distance method. The idea is also to confirm the distances using k-means built-in distance metrices such as Euclidean and Cityblock. The different distance calculation methods returns approximately the same results on short distances. We first run the k-means algorithm in Matlab to determine the locations of the local controllers, then use those locations to determine the location of the global controller.

TABLE 1 Sink Nodes and their Coordinates.

	Coordinates1	
Controller	**Latitude**	**Longitude**
1	-25.755584	28.278443
2	-25.755594	28.278934
3	-25.755668	28.278703
4	-25.756051	28.27778
5	-25.756131	28.278107
6	-25.756303	28.278242
7	-25.756447	28.279639
8	-25.756334	28.279703
9	-25.756331	28.279551
10	-25.756536	28.27827
11	-25.756528	28.278532
12	-25.756207	28.280012
13	-25.756572	28.280249
14	-25.755309	28.278177
15	-25.755426	28.278749

Source: Google Maps.

3.3 | Results and Discussion

The purpose of this exercise was to determine the best locations to place the local controllers and the global controller for the fragmentation model. The k-means method produces, in addition to average latency, the maximum latency (referred to by Heller *et al.*[24] as worst-case latency) and the distances of each point to the centroid. The locations of the three local controllers based on the k-means are depicted in figure 2 and listed in table 3 .

The main antagonists of the k-means algorithm argue that the placement is vulnerable to outliers, and as seen in figure 2 , the local controllers are placed closer to or on one of the sink nodes. Moreover, it is said that this poses a challenge in a very large network where the distances in between are very huge. But as in the case of the SDWSN, particularly the fragmentation model, this does not have a negative effect because the network is already fragmented. This is the reason we chose k-means, and furthermore, in a relatively small network such as the fragmented clusters, the different clustering techniques produce relatively similar results. This can be observed from the average value, the median value, and the maximum value produced.

TABLE 2 Coordinates and their weights.

Edge Connection	Weight (m)
(1,2)	49
(1,3)	28
(2,3)	25
(4,5)	34
(4,6)	54
(5,6)	23
(7,8)	14
(7,9)	16
(8,9)	15
(4,10)	73
(5,10)	48
(6,10)	26
(4,11)	92
(5,11)	61
(6,11)	38
(10,11)	26
(7,12)	46
(8,12)	34
(9,12)	48
(7,13)	63
(8,13)	61
(9,13)	75
(12,13)	47
(1,14)	41
(2,14)	82
(3,14)	66
(1,15)	35
(2,15)	26
(3,15)	27

TABLE 3 The locations of the local controllers.

	Coordinates1	
Controller	**Latitude**	**Longitude**
A	-25.7563	28.2782
B	-25.7563	28.2797
C	-25.7556	28.2787

On the placement of the global controller, we applied the k-means to the locations obtained when placing the local controllers above. This is highlighted in figure 3 below. As shown in the figure, the global controller is placed closer to the local controller closest to the mean.

Unlike the local controller placement, the global controller is expected to be at a relative distance to the local controllers and therefore placement closer to the mean would leave other observations vulnerable to high latencies. Therefore, we re-optimise the local controller locations to counter the error condition. The use of k-median is considered. However, it results in the same placement as the mean, as shown in figure 4 .

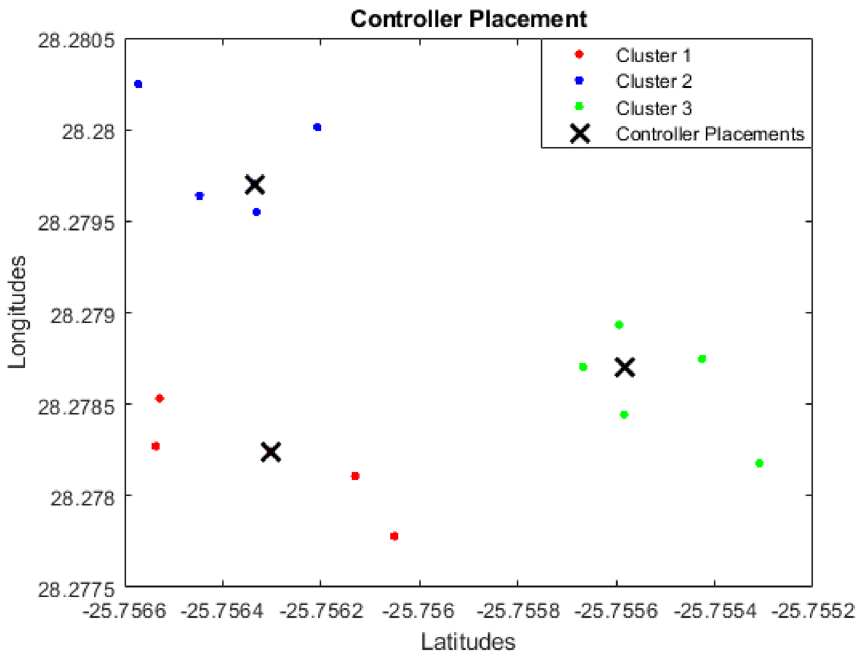


FIGURE 2 The locations of the local controllers by k-means algorithm.

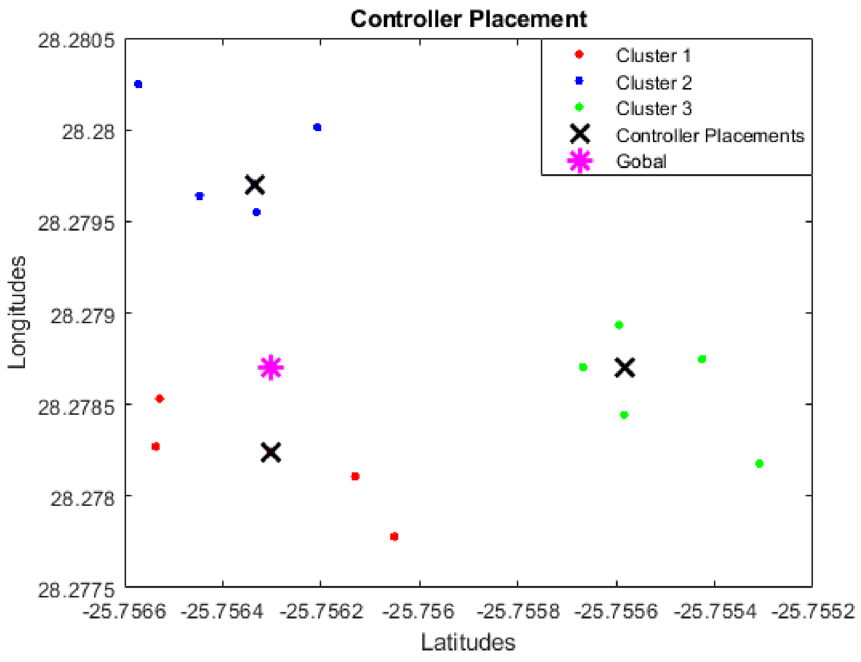


FIGURE 3 The locations of the global controller by k-means algorithm.

Increasing the topology from three to four or five local controllers (in case the topology grows) does not improve the positioning either. This means that the optimal placement of the global controller cannot rely on the increase in topology (data points) as shown in figures 5 a and 5 b. The mean would always be skewed and unpredictable. Besides the fact that this does not have much effect on centralising the global controller, this would be a bad method as it is not cost effective.

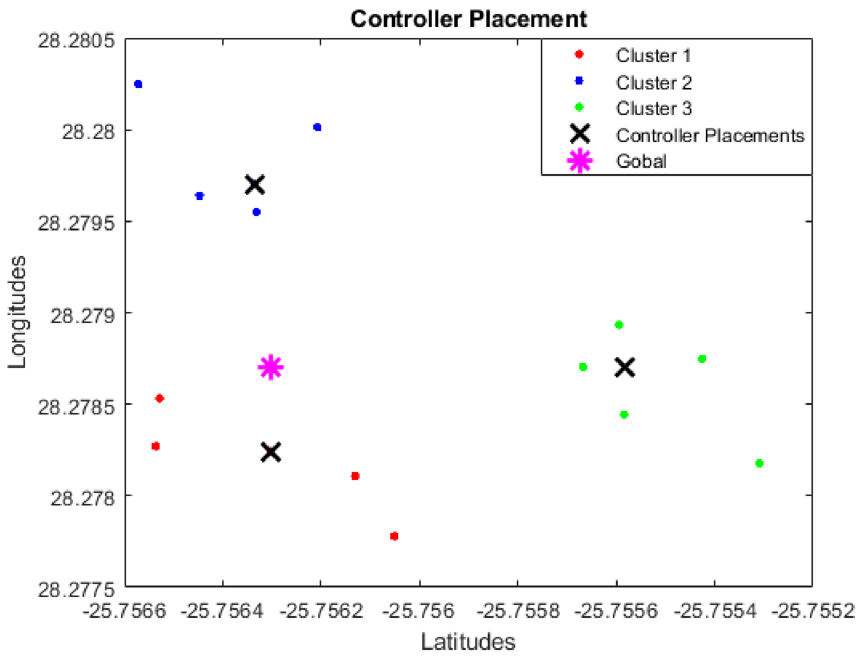


FIGURE 4 The use of the median is the same as that of the mean.

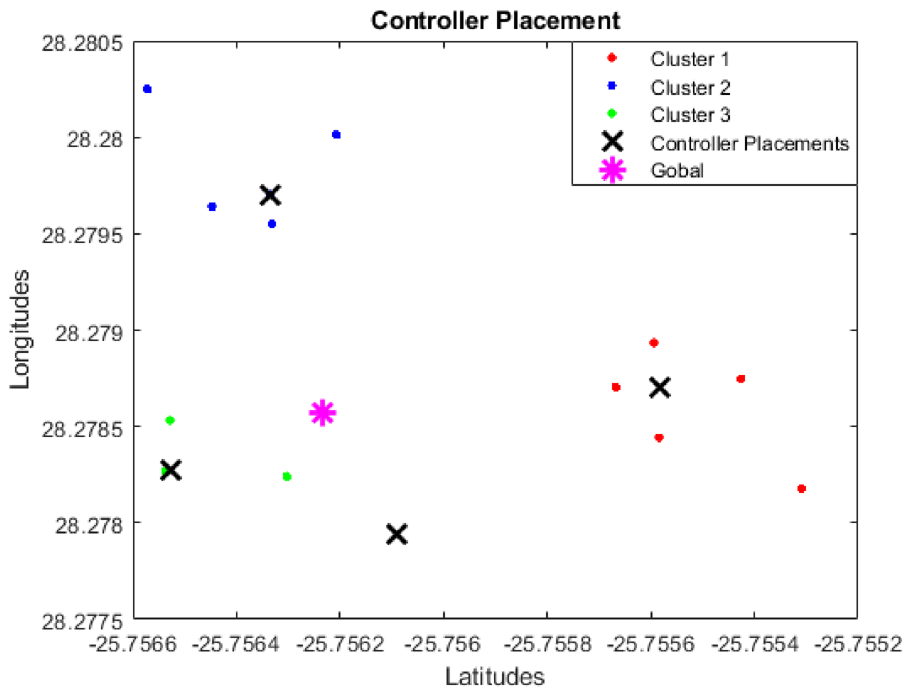
Therefore, we used the concept of moving median to re-optimise the locations of the local controllers to determine the optimal location of the global controller. Moving median computes a number of different k medians over a sliding window of the size of k. It shifts the window forward and backward over the size of k. It is mostly used to compute time series data. Table 4 lists the new locations derived from the moving median. The next step is to apply k-means to this new matrix to determine the new location of the global controller; we also find the median.

TABLE 4 The locations of the local controllers after applying moving median.

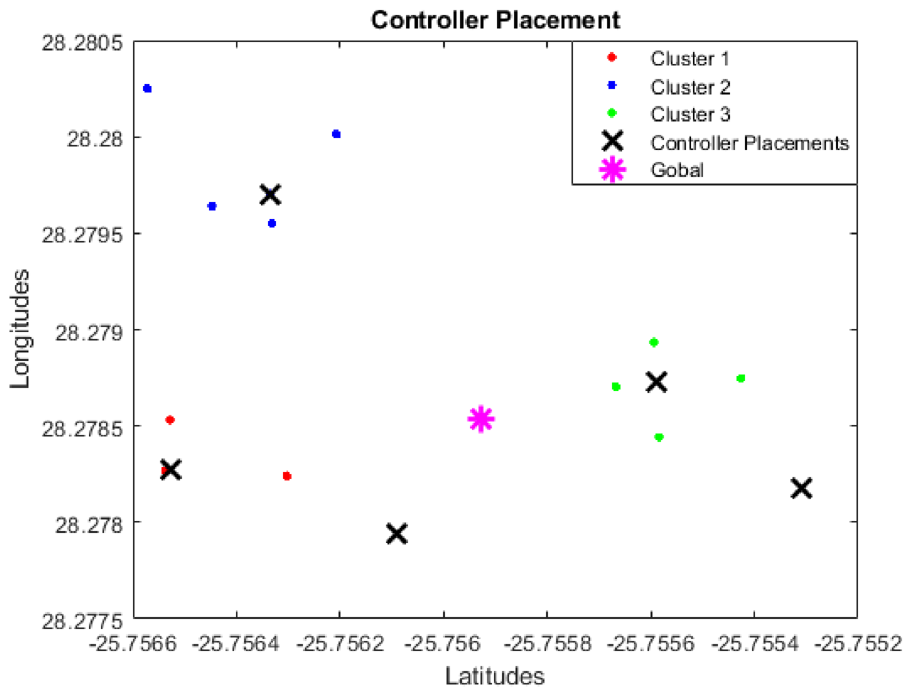
	Coordinates1	
Controller	**Latitude**	**Longitude**
A	-25.7563	28.2790
B	-25.7563	28.2787
C	-25.7560	28.2792

The new matrix above mimics the worst-case scenario described by Heller while conforming to the free fail scenario by Hock. These new locations move towards the k-center optimisation. However, noticeably, they cannot be used for the local controller placement as they converge towards the centre of the network. Figure 6 shows the locations of the local controller using the new matrix. Evidently, this is not suitable for the local controller placements according to the fragmentation ideals for SDWSN because as shown in figure 6 , they are further away from the sink nodes, which are their primary area of interest.

However, as these locations converge towards the centre of the network, they improve the placement location of the global controller. Figure 7 shows the location of the global controller with the mean and the median respectively. This results from applying the mean and median to the three locations obtained from moving median. The placement of the global controller improved greatly. There is a slight difference between the location of the mean and that of the median.



(a) The global controller placement with k = 4.



(b) The global controller placement with k = 5.

FIGURE 5 The increase in topology does not change the placement of the global controller when using k-means.

To find a better method between the mean and the median, we looked at the distances to each point. We took the cluster locations in tables 3 and 4 to determine the distance between each point and the centroid (global controller location) produced by the mean and median of the first matrix (table 3) and the mean and the median of the moving median matrix (table 4). We referred to the first matrix as S and the second matrix as S'. The mean and the median of the first matrix(S) were the same. We

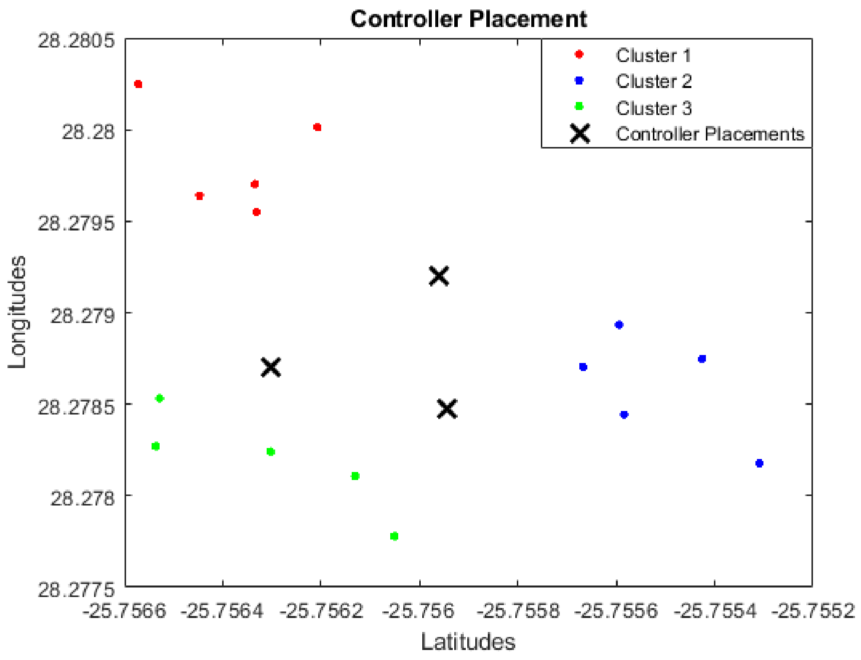


FIGURE 6 The locations of the local controller after re-optimisation.

used the Haversine method to calculate these distances using a different platform (Java code). The results are captured in table 5 .

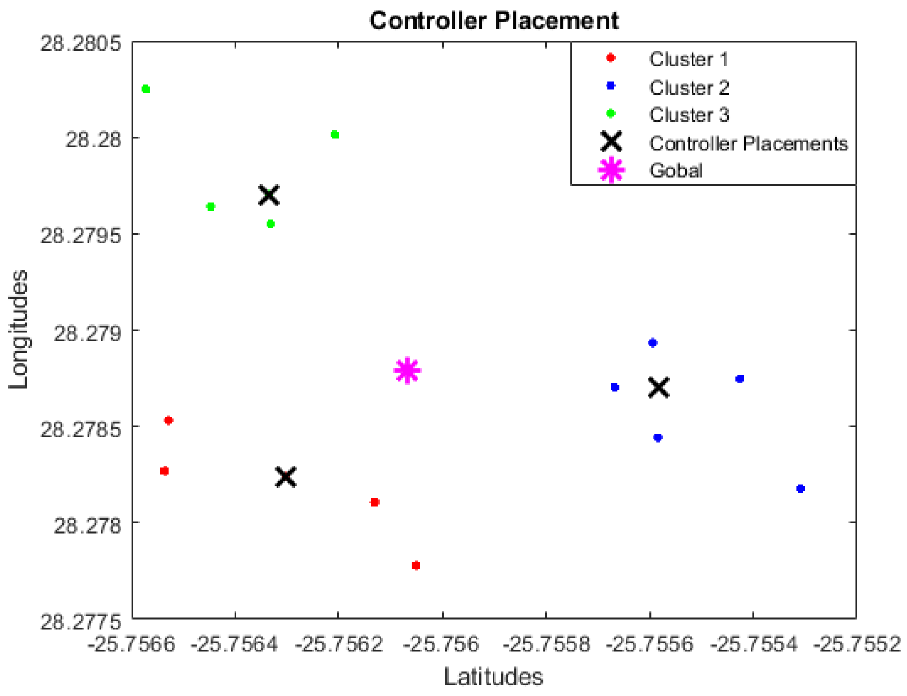
TABLE 5 The distances between the obtained locations and the centroids.

	Coordinates1		
Location	**S: mean/median**	**S':median**	**S':mean**
A	78	83	73
B	100	70	71
C	50	80	81

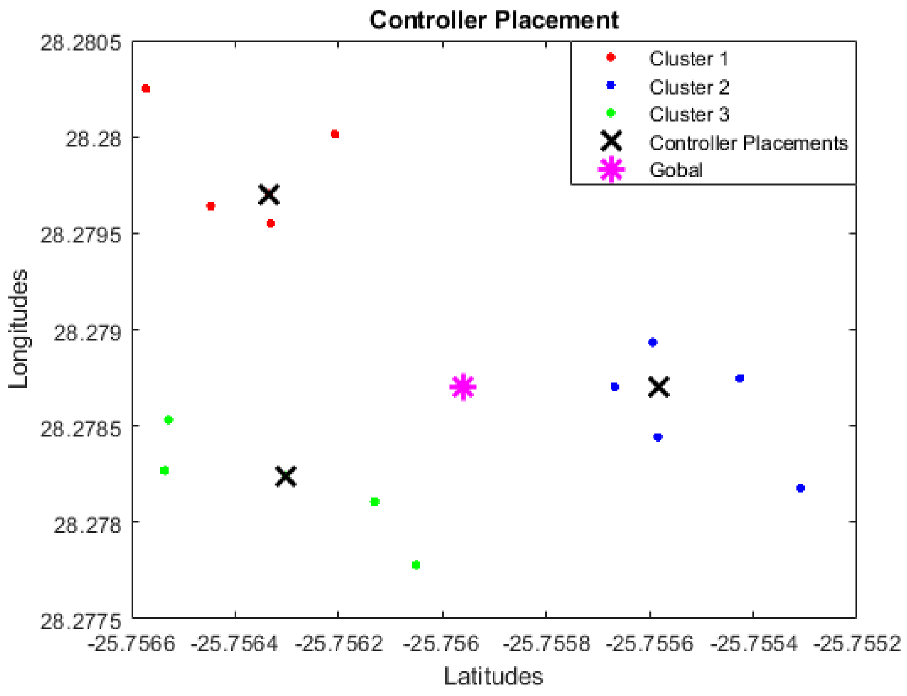
Table 5 shows that the mean has a better overall distance although the difference is slight. Both the mean and the median could therefore be used to determine the final location of the global controller. We can therefore conclude that the k-mean is suitable to place the local controller while the global controller should be placed by further optimising the local controller locations. This eventually converges to the k-center approach. The following algorithm 1 summarises the steps in placing the controller.

3.4 | Latency

The k-mean function in Matlab returns the best distances in the clusters. The average sums of the distances between the points in the cluster and the centroid (local controllers), measured in metres are 1.6, 1.6, and 1.8 for the three clusters respectively. The maximum distances between all the points to the centroids are 2.6, 2.5, and 2.3 in the three clusters. The average sums represent the k-mean value while the maximum points represent the k-center as described in Heller *et al.*[24] and Hock *et al.*[23]. The average



(a) The mean of the moving median locations.



(b) The median of the moving median locations.

FIGURE 7 The final optimisation of the locations to determine the global controller location.

distance between the local controllers and the global controller is 3.27. We use the standard distance, time, and speed equation to determine the latency in the form of time.

$$\begin{aligned} Distance &= Speed \times Time \\ \therefore Time &= \frac{Distance}{Speed} \end{aligned} \tag{8}$$

Algorithm 1 Controller placement

These are the steps to be followed:

- Place the local controllers using k-means or m-median clustering.
- Use the locations of the local controllers to determine the location of the global controller.
- Apply a moving median to the locations of the local controllers.
- Apply k-mean or k-median to the new derived locations to find the mean or the median for the location of the global controller.

We use the Ethernet over copper speed of 197863.022 ms. The resultant latencies are 8.0864, 8.0864, and 9.0972 ns respectively within the local clusters and 16.5266 ns between the local controllers and the global controller.

4 | CONTROLLER RE-ELECTION

Distributed systems use consensus algorithms to achieve state convergence. Consensus algorithms are normally discussed in terms of replicating the state amongst the participating controller nodes. Another perspective of the consensus algorithms is the election of a leader in a distributed system. The election of the leader ensures reliability in the system in the event of failure. It ensures that for any operation, there will be service and backup for assurance. As in database systems, consensus is very important for SDN, especially in distributed SDN controllers.

4.1 | ONOS Device Mastership

The mastership service in ONOS provides cluster management, synchronisation, and device mastership. The mastership service manages the device mastership. The ONOS device mastership management defines three roles[45,46]:

- Master: The controller node has knowledge of the device and has full control.
- Standby: The node has knowledge of the device, and can read the state, but cannot control the device.
- None: The device may or may not have knowledge of the device and cannot interact with it.

This is the ONOS mastership life cycle. A controller node becomes a master of a device if it discovers the device first and can verify that the device has no master and has a control channel to the device. All other nodes that subsequently discover the device become standby (if they have a connection) or None. Thus, in a case where a controller overlaps the sink sink node, it becomes on standby if it has a connection to that sink node. The roles can change through administrative intervention, device disconnection, and disconnection from the cluster (split-brain syndrome). All these will prompt a role relinquishment and node re-election to replace the master. Re-election can be the result of master node failure, device disconnection, and an administrator intervention. The node relinquishing the responsibilities can elect a new master. The candidate to be the new master is selected from the standby nodes. The standby nodes are ordered on a preference list and the next node on the list becomes the candidate, master select.

4.2 | SDWSN controller re-election

The fragmentation model distributes the controller instances across the network. In section 3, we discussed the best ways to consider when placing the local controllers. The main metric behind the controller placement problem is latency, which is modelled through distance. Therefore, if the initial placement takes distance into consideration, then the re-election of the master should do so too. However, currently ONOS does not consider distance in its device master re-election. In the traditional SDN, this might not be a concern, considering the available infrastructure resources. The SDWSN, unfortunately, does not possess the luxury of resources to spare, hence more consideration should be taken into account.

The current implementation contradicts the aims and ideals of the fragmentation model. This section aims to optimise the controller master re-election method. This entails enhancing the method with a distance consideration when a new master node is elected. Building upon the controller placement; we implement the Haversine formula of distance. This calculates the distance based on the latitude and longitude coordinates of the controllers. We gather the coordinates of the controller instances upon commissioning. We chose Haversine method because of its applicability and accuracy in long distances in anticipation of scalability because at short distances, most distance formulas return the same results. Thus, any distance method which uses latitudes and longitudes such as the law of cosine could be used. The pseudocode of the Haversine formula is given in algorithm 2 below.

Algorithm 2 The Haversine algorithm.

```
longitude lon1 longitude lon2 latitude lat1 latitude lat2
$R \leftarrow 6367000$ is the radius of the earth in metres.
$\Delta long \leftarrow lon2 - lon1$
$\Delta lat \leftarrow lat2 - lat1$
$\alpha \leftarrow (\sin(\frac{\Delta lat}{2}))^2 + \cos(lat1) \times \cos(lat2) \times (\sin(\frac{\Delta long}{2}))^2$
$\beta \leftarrow 2 \times \arctan(\sqrt{\alpha}, \sqrt{1-\alpha})$
$\gamma \leftarrow R \times \beta$
return $\gamma$
```

The local controllers should be at the centre of their clusters according to the placement criteria discussed in the previous section. We calculate the distances between the current master controller (the node relinquishing the role) and the standby controllers; the closest controller node becomes the candidate or the master select. The change of the master increases the latency, this exercise manages that increase by preferring the closest controller. The standard procedure for controller re-election after a controller node failure in ONOS is is described in algorithm 3:

Algorithm 3 Controller mastership re-election procedure in ONOS.

Upon a controller node failure:

- Relinquish the mastership role.
- The candidate node from the standby list is chosen as the replacement.

The standby list is populated as follows:

- First, the first controller to discover a device becomes the master controller, the rest become standby controllers for that device if they have a connection; they become none if they do not.
- All standby controllers are stored on a list in a first-come precedence order.
- The first controller on the list becomes the candidate.
- Upon controller failure, the candidate controller becomes the master.

The proposed controller mastership re-election after failure is described in algorithm 4 below. The proposed method can be proactive or reactive. The proactive mode will ensure a fast transition from a failed controller to the replacement controller, but it will add overheads. Whereas the reactive method will be slower the transit to the replacement controller but with no processing overheads. Although the differences will be very minimal, the proactive method is more suitable for delay-sensitive networks such as SDWSN.

Algorithm 4 Proposed controller mastership re-election procedure.

```
Upon a controller node failure:
  • Relinquish the mastership role.
  • The candidate node from the standby list is chosen as the replacement.
The procedure for the proposed enhancement changes the standby such that it:
  • Uses a HashMap to store the standby controllers with their coordinates.
  • Calculates the distance between the master and all the standby controllers, and stores the results in a HashMap.
      – Calls the Haversine distance method.
  • Selects the entry with the lowest distance as the candidate controller.
  • Upon Failure, the candidate becomes the master.
Haversine distance calculation method:
  • Find the latitude and longitude coordinates of both the current master controller and all the controllers in the cluster.
  • Measure the distance between the two coordinates. return distance
```

4.3 | Experiment

The controller master re-election enhancement was implemented on the ONOS mastership service. We set up a small experiment to test this enhancement. The experiment consisted of one global controller and three local controllers according to the fragmentation model. The three local controllers consisted of 2GHz CPU and 1G RAM while the global controller consisted of 2GHz CPU and 2G RAM. The global controller also ran the simulation tool. The simulation consisted of three sink nodes with ten sensor nodes each. All the controllers were virtual and placed in different geographical buildings at the Council for Scientific and Industrial Research (CSIR), Pretoria. Figure 8 depicts the buildings where the controllers were located (Picture from Google Maps).

We used the SDN-WISE simulation framework which consists of a Cooja simulator and an SDWSN adaptation of the ONOS SDN controller. The Fragmentation model was implemented in the ONOS controller on version 1.0.1.

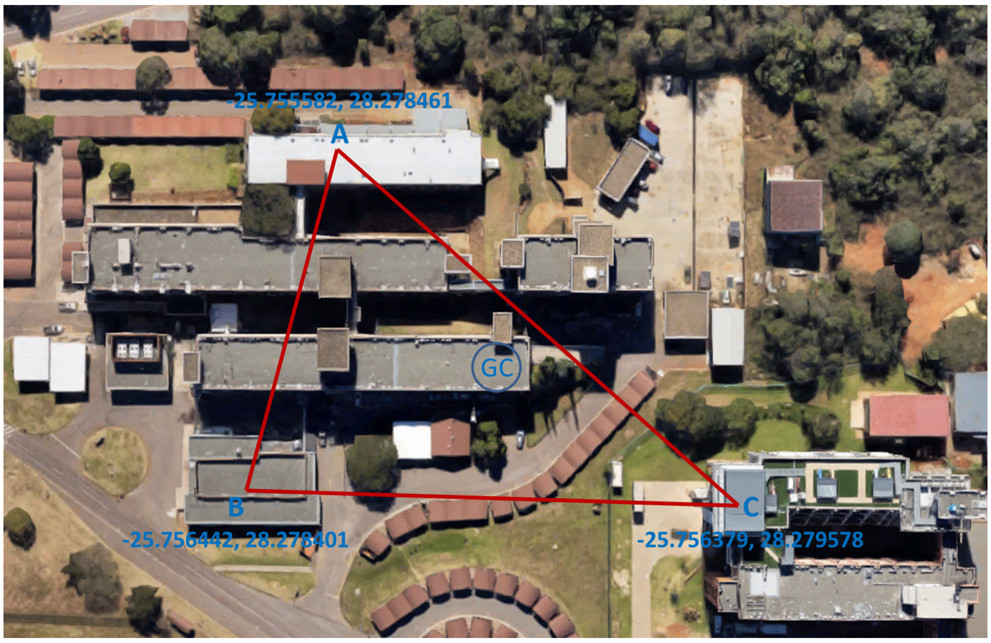


FIGURE 8 The location of the controllers in the buildings of CSIR, image from Google maps.

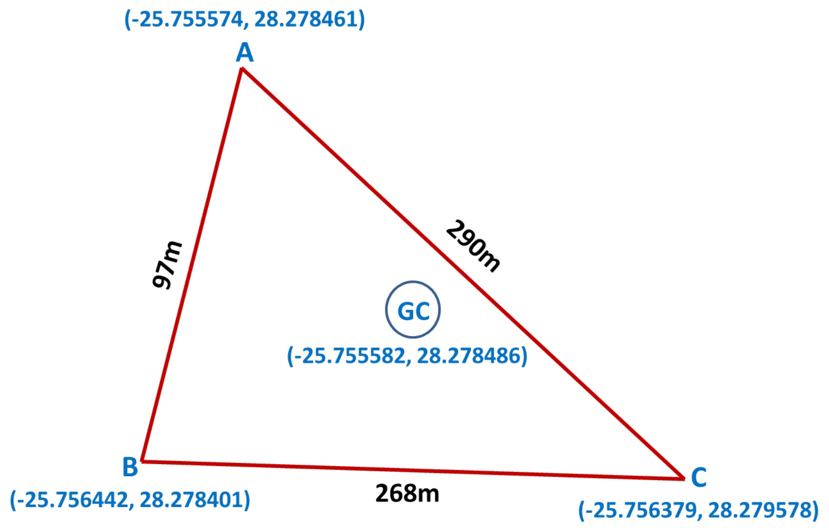


FIGURE 9 The geometric extraction of the controller locations.

The controllers were located in different buildings; figure 9 shows the latitudes and longitudes of the controllers' locations, as well as the distances in between. The central controller was also located, more in the centre according to the placement model discussed in section 3 above. The controllers and their location coordinates are summed in table 6 below.

TABLE 6 The local controllers and their location coordinates in latitude and longitudes.

	Coordinates1	
Controller	**Latitude**	**Longitude**
A	-25.755574	28.278461
B	-25.756442	28.278401
C	-25.756379	28.279578
GC	-25.755582	28.278486

Source: Google Maps.

We used a script provided by ONOS to manipulate the state of the controllers, we used "*onos-service*". This script allowed us to stop, start, restart, and check the status of a controller instance, thus "*onos stop/start/restart/status*". The testing procedure followed was:

1. Set up the cluster.
2. Verify that all controller instances are in ACTIVE state.
3. Run the SDWSN simulation.
4. Stop the one local controller instance in the middle of the simulation.
5. Verify if the stopped instance is indeed in INACTIVE state.
6. Confirm if the correct controller node was re-elected.

In the first experiment, we evaluated the proposed criterion by failing the local controllers one at a time. The second experiment centred around the current re-election criterion versus the proposed criterion in terms of latency. We measured the latency with the current criterion, then failed the controller and did the same with the proposed criterion. Controller B was used in this case. We measured latency on controller B, then failed it; then measured the latency with the replacement.

4.4 | Results and Discussion

The ONOS framework provides a command-line service which allows monitoring and manipulation. The setup depicted in figures 8 and 9 shows the different locations of the controllers with coordinates. The figures present a clear logical preference based on the different distances between the controllers, and thus the evaluation seeks to validate rather that test. The Haversine formula is also clear and we validated the formula away from ONOS to get the distances and used Google maps to verify them. The results obtained followed the logic as expected and presented in table 7 . Therefore, the table shows the results of the proposed re-election criteria. The proposed method chooses controller A to replace controller B and vice versa, while choosing controller B to replace controller C.

TABLE 7 The local controllers and their replacements.

Local Controller	Replacement	Distance (m)
A	B	97
B	A	97
C	B	268

Figure 10 below shows the results of the second experiment, the latency variations before the controller was failed and after. We performed all the failing on controller B, any controller could be used (failed) and that did not change the outcomes of the experiment. According to the results in table 7 , the proposed method will always choose the closest controller. However, the current method could choose any of the remaining two. The results in figure 10 below shows the latency before controller B is failed, then latency when the closest controller is chosen, and lastly latency when the furthest controller is chosen. The last two results are applicable to the current method because the choosing is random. In figure, 10 **B** refers to controller B before the failing, **B->A** refers to the replacement of controller B by controller A while **B->C** refers to the replacement of controller B by controller C.

The latency results presented show a slight addition of delay as the change of controllers takes place. Accordingly, the results show that the added delay is more when the controller replacement does not take distance into account. This is of significant importance in SDWSN. This difference could be deemed insignificant in traditional SDN, but very critical in SDWSNs. They are very important for SDWSNs because of their relativity and proportionality to scalability. Considering the distances between the controllers, the speed in traditional networks could overwrite the latency differences, but remains critical in SDWSNs.

A major challenge in SDWSN is that the concept is still in development stages and therefore there exist less tools which could be used. The simulation tool used has limitation in the number of sensor nodes it can simulate at a time. We could only simulate 39 sensor nodes, beyond which it crashed. A real world testbed could not be done because of the unavailability of SDN-based sensor nodes. However these challenges did not negate this research work because the focus was more on the controller redundancy mechanism.

5 | CONCLUSION

The Software Defined Wireless Sensor Networking (SDWSN) has ignited the Wireless Sensor Networks (WSN) in the recent times. It leverages the benefits of Software Defined Networking into WSNs. The distributed control systems for SDWSNs have

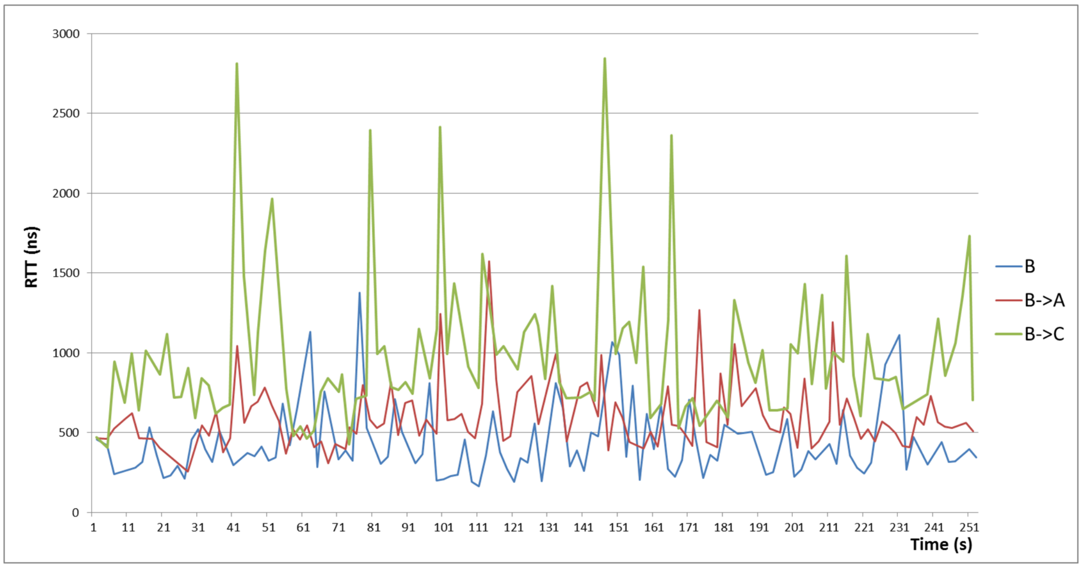


FIGURE 10 The latency of the two re-election criteria, the proposed and the current.

also gained traction and much research interest on the basis of their premise of improving reliability, scalability, and performance. Therefore, different distribution approaches have been proposed and one such is the Fragmentation model. It seeks entails a two level architecture which consists of local controllers and a global controller in consideration of the inherent ills of limited resources in SDWSNs. This paper serves as an enhancement of the fragmentation model in terms of latency and reliability. It seeks to strengthen and optimise the fragmentation model for real life operation. We discussed in detail two problems, the controller placement and a controller mastership re-election. The controller placement seeks to reduce the propagation latency between the sink nodes and the local controllers, as well as between the local controllers and the global controller. The experiments showed that we cannot use the same method of optimal placement for both the local controllers and the global controller. Therefore two models of optimal controller placement were discussed and adopted. K-means for the local controller placement and re-optimised k-means or k-center for the global controller placement. We also enhanced the current ONOS mastership re-election criterion with distance consideration in line with the ideals of the fragmentation model. We adopted the Haversine formula and implemented it in ONOS to ensure that the re-election was in line with the fragmentation model. These two problems are intertwined and complementary. This ensure that propagation latency is kept to a minimum and, at an acceptable threshold after a controller failure. The change of controller inevitably adds to the latency and the role of this exercise is to ensure that even after failure, the network does not suffer abnormal latencies to the detriment of the network.

ACKNOWLEDGMENTS

This work is supported by the Council for Scientific and Industrial Research (CSIR). Opinions expressed and conclusions arrived at are solely those of the authors and should not be attributed to either the University of Pretoria or the Council for Scientific and Industrial Research.

6 | ORCID

Hlabishi Kobo https://orcid.org/0000-0003-4362-2363

References

1. Kobo Hlabishi I., Abu-Mahfouz Adnan M., Hancke Gerhard P.. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access.* 2017;5(2):1872–1899.
2. Kreutz Diego, Ramos Fernando M. V., Esteves Verissimo Paulo, Esteve Rothenberg Christian, Azodolmolky Siamak, Uhlig Steve. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE.* 2015;103(1):14–76.
3. Benzekki Kamal, El Fergougui Abdeslam, Elbelrhiti Elalaoui Abdelbaki. Software-defined networking (SDN): a survey. *Security and Communication Networks.* 2016;9(18):5803–5833.
4. Akpakwu G.A., Silva B.J., Hancke G.P., Abu-Mahfouz A.M.. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access.* 2017;6(12):3619 – 3647.
5. Ogbodo Emmanuel U., Dorrell David, Abu-Mahfouz Adnan M.. Cognitive Radio Based Sensor Network in Smart Grid: Architectures, Applications and Communication Technologies. *IEEE Access.* 2017;5(8):19084–19098.
6. Yaseen Qussai, Albalas Firas, Jararwah Yaser, Al-Ayyoub Mahmoud. Leveraging fog computing and software defined systems for selective forwarding attacks detection in mobile wireless sensor networks. *Transactions on Emerging Telecommunications Technologies.* 2018;29(4):e3183.
7. Cheng Bo, Zhang Jingyi, Hancke Gerhard P., Karnouskos Stamatis, Colombo Armando Walter. Industrial Cyberphysical Systems: Realizing Cloud-Based Big Data Infrastructures. *IEEE Industrial Electronics Magazine.* 2018;12(1):25–35.
8. Miyaji Atsuko, Omote Kazumasa. Self-healing wireless sensor networks. *Concurrency and Computation: Practice and Experience.* 2015;27(10):2547–2568.
9. Huebner Christof, Cardell-Oliver Rachel, Hanelt Stefan, Wagenknecht Tino, Monsalve Alvaro. Long-range wireless sensor networks with transmit-only nodes and software-defined receivers. *Wireless Communications and Mobile Computing.* 2013;13(17):1499–1510.
10. Omolemo G.M., Abu-Mouhfaz A.M.. Utilising Artificial Intelligence in Software Defined Wireless Sensor Network. In: :6131–6136; 2017; Beijing.
11. Ndiaye Musa, Hancke Gerhard P., Abu-Mahfouz A. M.. Software Defined Networking for Improved Wireless Sensor Network Management : A Survey. *Sensors.* 2017;17(5):1–32.
12. Kobo Hlabishi I., Hancke Gerhard P., Abu-Mahfouz Adnan M.. Towards A Distributed Control System For Software Defined Wireless Sensor Networks. In: :6125–6130; 2017.
13. Pritchard S.W., Hancke G.P., Abu-Mahfouz A.M.. Security in Software-Defined Wireless Sensor Networks: Threats, challenges and potential solutions. In: :168 – 173; 2017; Emden.
14. Berde Pankaj, Snow William, Parulkar Guru, et al. ONOS: Towards an Open, Distributed SDN OS. In: :1–6; 2014.
15. Medved Jan, Varga Robert, Tkacik Anton, Gray Ken. OpenDaylight: Towards a Model-Driven SDN Controller architecture. In: :1–6; 2014.
16. Tootoonchian Amin, Ganjali Yashar. HyperFlow: a distributed control plane for OpenFlow. In: :1–6; 2010.
17. Yeganeh Soheil Hassas, Ganjali Y. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In: :19–24; 2012.
18. Dixit Advait Abhay, Hao Fang, Mukherjee Sarit, Lakshman T.V., Kompella Ramana. ElastiCon: An Elastic Distributed SDN Controller. In: :17–28; 2014.
19. Galluccio Laura, Milardo Sebastiano, Morabito Giacomo, Palazzo Sergio. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In: :513–521; 2015.

20. Oliveira Bruno Trevizan, Margi Cintia Borges, Gabriel Lucas Batista. TinySDN: Enabling Multiple Controllers for Software-Defined Wireless Sensor Networks. In: :1–6; 2014.

21. Oliveira Bruno Trevizan, Margi Cintia Borges. Distributed Control Plane Architecture for Software-defined Wireless Sensor Networks. In: :85–86; 2016.

22. Kobo Hlabishi Isaac, Abu-Mahfouz Adnan M., Hancke Gerhard P.. Fragmentation-based Distributed Control System for Software Defined Wireless Sensor Networks. *IEEE Industrial Informatics.* 2018;In Press.

23. Hock David, Hartmann Matthias, Gebert Steffen, Jarschel Michael, Zinner Thomas, Tran-Gia Phuoc. Pareto-optimal resilient controller placement in SDN-based core networks. In: :1–9; 2013.

24. Heller Brandon, Sherwood Rob, McKeown Nick. The Controller Placement Problem. In: :7–12; 2012.

25. Whelan Christopher, Harrell Greg, Wang Jin. Understanding the K-Medians Problem. In: :219–222; 2015.

26. Arya Vijay, Garg Naveen, Khandekar Rohit, Munagala Kamesh, Pandit Vinayaka, Pandit Vinayaka. Local Search Heuristic for k-median and Facility Location Problems. In: :21–29; 2001.

27. Williamson David P., Shmoys David B.. *The Design of Approximation Algorithms - David P. Williamson, David B. Shmoys - Google Books.* Cambridge University Press; 1st ed.2011.

28. Hock David, Hartmann Matthias, Gebert Steffen, Zinner Thomas, Tran-Gia Phuoc. POCO-PLC: Enabling dynamic pareto-optimal resilient controller placement in SDN networks. In: :115–116; 2014.

29. Lange Stanislav, Gebert Steffen, Zinner Thomas, et al. Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks. *IEEE Transactions on Network and Service Management.* 2015;12(1):4–17.

30. Bari Md. Faizul, Roy Arup Raton, Chowdhury Shihabur Rahman, et al. Dynamic Controller Provisioning in Software Defined Networks. In: :18–25; 2013.

31. Lin Shih-Chun, Wang Pu, Akyildiz Ian, Luo Min. Towards Optimal Network Planning for Software-Defined Networks. *IEEE Transactions on Mobile Computing (Early Access).* 2018;:1–1.

32. Ishigaki Genya, Gour Riti, Yousefpour Ashkan, Shinomiya Norihiko, Jue Jason P.. Cluster Leader Election Problem for Distributed Controller Placement in SDN. In: :1–6; 2017; Singapore, Singapore.

33. Muqaddas Abubakar Siddique, Bianco Andrea, Giaccone Paolo, Maier Guido. Inter-controller traffic in ONOS clusters for SDN networks. In: :1–6; 2016.

34. Faragardi Hamid Reza, Fotouhi Hossein, Nolte Thomas, Rahmani Rahim. A Cost Efficient Design of a Multi-sink Multi-controller WSN in a Smart Factory. In: :594–602; 2017.

35. Lamport Leslie, Leslie . The Part-time Parliament. *ACM Transactions on Computer Systems.* 1998;16(2):133–169.

36. Lamport Leslie. Paxos Made Simple. *ACM SIGACT News.* 2001;32(4):51–58.

37. Ongaro Diego, Ousterhout John. In Search of an Understandable Consensus Algorithm. In: USENIX ATC'14:305–320; 2014.

38. Zhang Yang, Ramadan Eman, Mekky Hesham, Zhang Zhi-Li. When Raft Meets SDN. In: :1–7; 2017.

39. Hunt Patrick, Konar Mahadev, Junqueira Flavio P., Reed Benjamin. ZooKeeper: Wait-free Coordination for Internet-scale Systems. In: :1–11; 2010.

40. Hazelcast . Hazelcast the Leading In-Memory Data Grid - Hazelcast.com .

41. Atomix . Atomix - Fault-tolerant Distributed Coordination Framework for Java 8 .

42. Mostafaei Habib, Menth Michael. Software-defined wireless sensor networks: A survey. *Journal of Network and Computer Applications.* 2018;119(10):42–56.

43. Abu-Mahfouz A. M., Hancke G.P.. Localised Information Fusion Techniques for Location Discovery in Wireless Sensor Networks. *International Journal of of Sensor Networks (IJSNET).* 2017;26(1):12–25.

44. k-means Clustering - MATLAB kmeans .

45. Beck Michael Till, Werner Martin, Feld Sebastian, Schimper Thomas. Mobile Edge Computing : A Taxonomy. In: :48–54; 2014.

46. ONOS . ONOS .

AUTHOR BIOGRAPHY

Hlabishi I. Kobo. Hlabishi Kobo received his Bachelor and Masters of Science in Computer Science from the University of the Western Cape in 2010 and 2012 respectively and recently received his PhD in computer engineering from University of Pretoria (2018). He worked for Telkom SA as a technology architect. He is currently a senior research engineer at the Council for Scientific and Industrial Research (CSIR). His main interests are software defined networking, software defined wireless sensor networks and software architecture.

Adnan M. Abu-Mahfouz. Dr Adnan M. Abu-Mahfouz (M'12-SM'17) received his M.Eng. and Ph.D. degrees in computer engineering from the University of Pretoria. He is currently a Principal Researcher with the Council for Scientific and Industrial Research, Research and Innovation Associate, Tshwane University of Technology, and also an extraordinary Faculty Member with the University of Pretoria. He participated in the formulation of many large and multidisciplinary R&D successful proposals. His research interests are wireless sensor and actuator network, low power wide area networks, software defined wireless sensor network, cognitive radio, network security, network management, and sensor/actuator node development. He is a member of many IEEE Technical Communities. He is an Associate Editor at the IEEE Access, the IEEE Internet of Things, and the IEEE Transactions on Industrial Informatics. He is the Founder of the Smart Networks collaboration initiative that aims to develop efficient and secure networks for the future smart systems, such as smart cities, smart grid, and smart water grid.

Gerhard P. Hancke. Dr Gerhard Hancke is currently Assistant Professor with Department of Computer Science at City University of Hong Kong. He received a Bachelor and Masters of Engineering degrees in Computer Engineering from the University of Pretoria (South Africa) in 2002 and 2003, and a PhD in Computer Science for the Security group at the University of Cambridge's Computer Laboratory in 2008. He worked for the Smart Card Centre and Information Security Group at Royal Holloway, University of London. His main interests are sensing applications and security of embedded system.

How to cite this article: H.I Kobo, A.M Abu-Mafhouz, and G.P Hancke (2019), Efficient Controller Placement and Resilience Mechanism in Distributed Control System for Software Defined Wireless Sensor Networks, *Int. JCommun. Syst.*, *2019;*.