A Novel Controller Placement Using Petri-Nets for SDNs

WAEL HOSNY FOUAD ALY
College of Engineering and Technology
American University of the Middle East
KUWAIT

Abstract: - Software defined networking (SDN) separates the control and the data planes. This separation brings flexibility to the network. But the decoupling has some drawbacks such as the controller placement problem (CPP). Controller placement is a crucial task which affects the overall networks' performance. This paper proposes a novel controller placement model that is based on petri-nets to place the SDN's controllers. The proposed model is called controller placement using petri-nets for SDNs ($CPPN_{SDN}$). $CPPN_{SDN}$ aims to reduce the average propagation latency among switches and their associated controllers. $CPPN_{SDN}$ divides the network into sub-networks. Each sub-network is governed by a controller. Experiments were conducted on the Internet2/OS3E topology to evaluate the performance of $CPPN_{SDN}$. Experiments show that $CPPN_{SDN}$ reduces the average latency significantly compared to two reference models. The first reference model is the *Modified Density Peaks Clustering (MDPC)* and the *Optimized* K_{means} model. In terms of the overall average latency, the $CPPN_{SDN}$ has shown promising results as it outperformed the MDPC and optimized K_{means} reference models by 7% and 17% respectively. Confidence Interval (CI) used was 90%. This is an ongoing work and the results are promising for more future investigation.

Key-Words: - Software-defined network; controller placement problem; density peaks clustering; petri-nets; modified density peaks clustering; optimized K_{means}.

Received: June 1, 2020. Revised: October 13, 2020. Accepted: November 9, 2020. Published: December 8, 2020.

1 Introduction

Software-defined networking (SDN) design started by having one centralized controller that manages the entire network. Single controller based architectures are usually appropriate for small LAN networks. WANs cover larger areas with different propagation delays and hence single controller architectures are not adequate for WANs due to scalability constraints. For that reason multi-controllers based architectures are suitable for WANs where the network is divided into sub-networks. Each sub-network is assigned a single controller.

Petri-net modeling is one of the most powerful modeling tools that facilitates modeling both mathematically and graphically for various network designs [1]. This paper proposes a novel petri-net framework that is mathematically proven to be valid for SDN architectures. The proposed model is called controller placement using petri-nets for SDN networks ($CPPN_{SDN}$). $CPPN_{SDN}$ adapts towards different types of controllers in a CPP. Typically, a controller could be a master controller or a slave controller. $CPPN_{SDN}$ uses a mathematical model that is based on petri-net frameworks to define different SDN network components. $CPPN_{SDN}$ is evaluated through computing the network's overall average and worse-case propagation delays. These values are compared to two reference models namely *modified density peaks clustering (MDPC)* and *optimized* K_{means}. In this work, the controller *propagation latency* is defined as the time that an incoming request takes at the ingress switch until the new data plane rules are established in the appropriate switch. It is assumed that the switches are responsible for the data forwarding. Switches forward various data based on flow rules. Switches request flow rules from the associated controller. Since the switches and their associated controllers have different distances, various propagation delays are observed for different flow rules. Since the propagation delays drastically vary among different controllers based on their physical locations, there is a desperate need to carefully place the controllers. Controllers' placement should aim to minimize the overall average propagation latencies.

The paper is organized as follows: Section 2 has the related work. Section 3 has the basic idea to model SDNs using the petri-net concept. Section 4 has the Modified density peaks clustering (MDPC) and optimized K_{means} reference models. Section 5 has the controller placement using petri-nets for SDNs ($CPPN_{SDN}$) proposed model. Section 6 has the experimental results. Section 7 has the conclusion and the future work.

2 Related Work

This section has a short literature survey about different modelling techniques that are used for the controller placement problem (CPP). CPP was first discussed by Heller et al. [2] where the CPP was modelled as a *K-mean* problem. Since the CPP has an influence on the propagation delays, authors defined a set of metrics that are considered as the base of the research in the CPP. Guo et al. [3] studied both network latency and network reliability through a solution that is based on greedy algorithm.

The proposed solution proved to outperform the reliability of similar approaches. Bari et al. [4] modelled the CPP as a backpack problem using greedy-knapsack model. The cost function used was the reciprocal of the cost path among the controllers and their associated switches. Loops were used as cost functions to find the controllers' coordinates. Zhang et al. [5] proposed a used the minimum-cut algorithm to model the CPP. The minimum-cut algorithm breaks the network into a set of clusters. Each subcluster contains two or more nodes. The number of clusters used is the number of controllers. Each subcluster is considered as the controller collision domain. MacQueen et al. [6] modelled the controller placement problem using integer programming models. Authors used the percentage of control path loss as the performance metric. Jalili et al. [7] modelled the CPP problem as a multi-objective programming problem. The model uses a compromising technique with different performance constraints where the network was sub-divided into clusters.

Aly et al. [8] worked on applying petri-net frameworks to improve the SDN fault tolerance. The approach used was successfully proven to be sound and gave promising result to increase the reliability of the controllers in SDNs when comparing the results to reference models. Using petri-nets to improve fault tolerance of controllers in SDN environment was promising and encouraging. Aly et al. [9][10] proposed a feedback control theoretic techniques to implement fault tolerance for controllers. The work gave promising results, but the feedback control theoretic techniques have put extra burden on the controllers. The ECFT [11] introduced load balancing at controller's failure, the proposed ECFT model focuses on balancing the load among other neighboring controllers. The proposed ECFT uses only delay among switches and their associated controllers in order to compute the load for each neighbor controller and sort the slave controllers accordingly

Qi et al. [1] proposed a model called *modified density peaks clustering (MDPC)*. MDPC uses the density of switches to partition a large network into several single controller sub-networks. Due to the spacing between the controllers and their associated switches, a large variety of propagation latencies are observed during the various requests of flow rules. Authors observed a correlation between the placement of the controller and the propagation delay between the controllers and their associated switches. Different values for the propagation delays have an important influence on the overall network performance.

3 Modelling using Petri-Nets

The section starts with a brief background about petri-nets and their different architectures to model the SDN networks.

3.1 Background about Petri-nets

In recent years, many network systems have been modeled using petri-nets. Petri-nets provide graphical representations for distributed systems [12]. For that reason petri-nets are used to model systems in which synchronization, communication and resource sharing are significant. Petri-nets have computer tools that support various design, simulation, and performance analysis of petri-net models [13]. Petri-nets are scalable and used to design and implement various systems without considering the size of these systems. Petri-nets have the ability to provide accurate demonstration of the behavior and the structure of modeling dynamic and transitional systems. They are particularly attractive for capturing features such as concurrency, asynchronous operation, synchronization, and flow of control. Once a petri-net model of a system is created, it can be utilized in a variety of ways [1].

Petri-nets are directed graphs that are mainly composed of two elements. The first element is the *transition element* Transition elements are represented graphically by rectangles. The second element is the *place element*. Place elements are represented graphically by circles. A place element could be connected to one or more transition elements, and a transition element could also be connected to one or more place elements. Fig. 1 shows the place element when it is connected to one transition element. As shown in Fig. 1, the solid circles are *tokens*. Tokens represent the activities

performed by the transition elements. Activities reside in places [14].

An empty place element that is connected to a transition element could disable a transition element from being executed. A transition element is said to be *enabled* if and only if there is no empty place elements connected to it. A transition element could "fire" after being enabled. The result of the firing process could be through the removal of the tokens from each of the transitions' input. This process could result in creating tokens in each of the output places. Arcs are labeled with the positive integers called *weights*. Each place element contains nonnegative tokens. The distribution of token elements over the places represents a configuration of the net called the *marking* [15]. Petri-nets modelling was successfully used in the field of SDN fault tolerance [8].

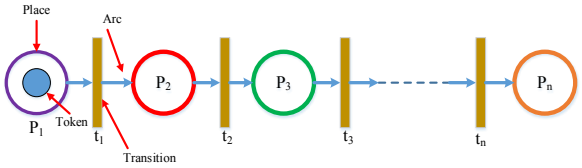

Fig. 1: Petri-net concept

A transition element is able to fire a token if each input place of the transition contains at least one token and the number of tokens is not less than the weight of the arc from the place to the transition. Firing occurs while the transition is enabled. It depends on whether the event actually takes place.

3.2 Modelling SDN Architectures using Petri-Nets

This section discusses the modeling of different SDN architectures using petri-nets. Three architectures are presented: (1) Single SDN controller, (2) multiple master SDN controller, and (3) hierarchical SDN controller architecture models.

3.2.1 Single SDN Controller Architecture

This section has the single controller architecture using petri-net components as shown in Fig. 2. The model has a single master controller where a set of switches are connected to it. Computation of the forwarding path is performed at the master controller. The computation depends on the flow request. Moreover, the master controller updates the switches by sending entries to the flow tables. Subsequently, packets of the incoming flow are forwarded based on the values of computed forwarding decisions.

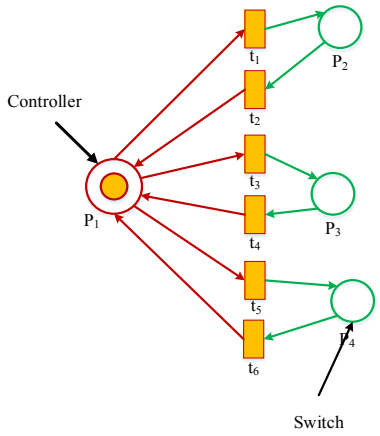

Fig. 2: Single SDN Controller Model

3.2.2 Multiple Master SDN controller Model

Multiple master SDN model is an example of a distributed architecture. This architecture consists of a set of master-controllers that communicate through message passing technique as shown in Fig. 3. In this model, switches are connected directly to their associated master-controllers.

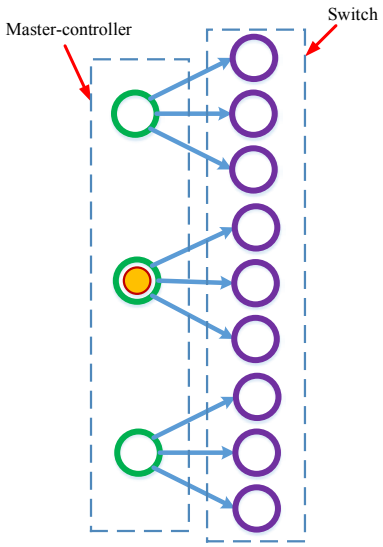

Fig. 3: Distributed SDN Controllers Model

3.2.3 Hierarchical SDN Controller Model

The hierarchical SDN controller architecture is a centralized model where there is more than one level of nodes. The model is composed of a master-controller and a set of slave-controllers that are connected to the master-controller. Switches are connected to the slave controllers as shown in Fig. 4. Network components whether they are switches or controllers are represented as circles.

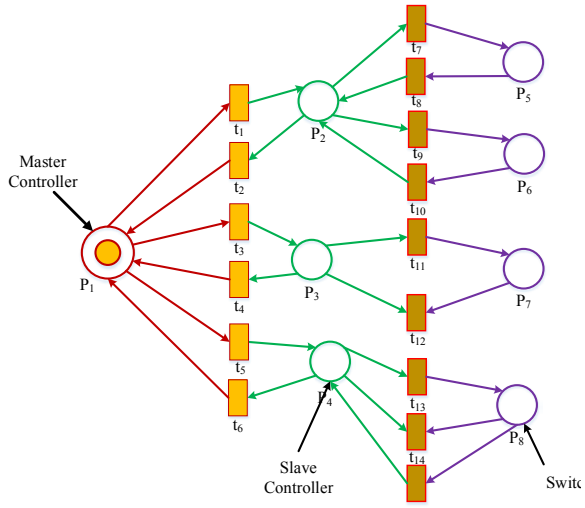

Fig. 4: Hierarchical SDN Controller Model

4 Reference Models

This section has the reference models used in this paper. The first subsection discusses the *optimized K-mean* algorithm.

4.1 Optimized K-median Algorithm

The first reference model is described by Wang et al. [17]. The model is called *optimized K-median* algorithm. The optimized *K-median* algorithm subdivides the network into group of clusters. Authors in [17] have divided the longest path in each partition to reduce the propagation delays for each sub-network. The optimized K-median experimental results show that the worst-case latency is given better results when compared to classical K-median by MacQueen et al. [6]. Wang et al. [17] measured the distance between the two nodes using Euclidean method rather than the physical link distance used in the classical K-median algorithm.

4.2 Modified Density Peaks Clustering (MDPC)

The second reference model used in this paper is called the *modified density peaks clustering (MDPC)* by Qi et al. [1]. MDPC is built on top of the classical density peaks clustering technique. The classical density peaks clustering technique is discussed by Rodriguez et al. [5]. It relies on two assumptions. The first assumption states that the SDN clusters' centres have relatively high local densities compared to non-centers. The second assumption states that the cluster centers are located in large dense compared to other non-centric points.

MDPC model clusters the network components into multiple areas. Switches are assumed to be connected to their associated controllers. MDPC uses metrics such as the *average degree parameter* and *closeness centrality parameter* to ensure controllers' centricity. In the MDPC model, the SDN is denoted by the undirected graph $\gamma = (\upsilon, \varepsilon)$ [1], where υ represents the set of switches, and ε represents the set of physical links. $\eta = |\upsilon|$ is the number of the switches, κ denotes the number of controllers. ρ $(\upsilon_i; \varepsilon_i)$ denotes cluster network. υ_i represents the set of switches in a sub-network i, while ε_i denotes the set of physical links in sub-network i. MDPC uses the density peak clustering based density. The density peak clustering depends on giving two quantities for each point, (1) local density and (2) distance to high local density point. MDPC assumes that the cluster centres density is higher than non-centric locations. MDPC takes topology $\Phi = (\upsilon, \varepsilon)$ as input topology. MDPC calculates the distance among all inputs to be able to compute the local density. MDPC then calculates the largest distance. According to the graph produced MDPC selects clusters' centres; which have the largest density and distance.

5 Controller Placement using Petri-Nets for SDNs (CPPNSDN)

This section has the proposed model which is referred to in this paper as *controller placement using petri-nets for SDNs* ($CPPN_{SDN}$). The first subsection introduces important terms that are going to be used while describing the $CPPN_{SDN}$. The following subsections discuss the theorem of soundness then a proof is provided to show the conditions under which the $CPPN_{SDN}$ framework is considered to be sound.

5.1 Petri-net Terms & Definitions

This subsection has the terms and definitions that are used throughout the $CPPN_{SDN}$ description in this paper. Each network component is assumed to have a set of capabilities. $CPPN_{SDN}$ validate the soundness of each network component and verify the model mathematically.

$CPPN_{SDN}$ is described by the tuple $\langle \aleph, \Psi, \Theta, \Phi \rangle$ where $\aleph$ is the set of nodes in the network, Ψ is the set of the network components' capabilities. The capabilities are defined as $\Psi=\{\Psi_1, \Psi_2, \Psi_3, \ldots \Psi_\omega\}$

where ω is the number of capabilities, Θ is defined as the network task coverage. The task coverage is the capability of the system to comply with the requested requirements of a given task. The topology used is defined by the symbol Φ.

Assume that there are three different types of capabilities for the network elements such as capacity, propagation delay, packet loss, etc. The task coverage is computed through the task coverage matrix.

$$\aleph \begin{matrix} \\ \\ \\ \end{matrix} \overset{\Psi}{\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}$$

The task coverage matrix shows the cross product of ℵ and Ψ. The cross product tests if the node covers the task specified parameter by setting the boolean value of the parameter with either 1 (satisfied) or 0 (unsatisfied).

For example for if the Internet2 OS3E topology [19] is used. The Internet2 OS3E topology has 34 nodes and 42 links as shown in Fig. 5. Each node is assumed to have a set of different capabilities. The matrix in this example has 30 rows. Assuming that the nodes have three capabilities (capacity, response time, packet loss). In that example, the matrix has three columns. The first of the matrix [1 1 0] indicates that the first node (switch) has high capacity and low propagation delay, but the packet loss is high. Similarly this strategy holds for the remaining 33 nodes of the topology Φ.

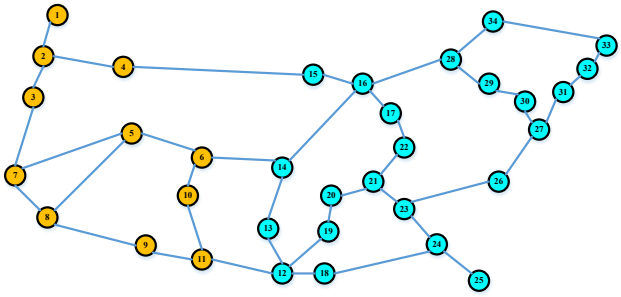

Fig. 5: Internet2 OS3E topology

By utilizing the concepts of petri-nets discussed above to the SDN components, for $CPPN_{SDN}$ framework, the places are considered to be either controllers (master and/or slave), or switches. Transitions are assumed to be the process of transmitting data from one node to the other. Every transition has a delay that depends on the communication parameters and the latency between the input and the output nodes.

5.2 Soundness Theorem

According to the proposed model, $CPPN_{SDN}$, an SDN is considered to be sound if and only if each network component has its task coverage satisfied. $CPPN_{SDN}$ defines the set of controllers mathematically as $X = \{\chi_1, \chi_2, \ldots \chi_k\}$. The cross product of the set of controllers in the SDN is defined by Ω. Therefore, $\Omega = X \times X$. X is considered to be *sound* if and only if $\exists\Omega = X \times X$, such that $\Omega \neq \phi$. X is distributed in an Internet2 OS3E topology Φ. As seen in Fig. 5 there are not any unreachable controllers. $CPPN_{SDN}$ could mathematically represent that the controllers are not unreachable as $\forall\chi_i, \chi_j, \chi_i \in\Omega$ and $\chi_j\in\Omega$, $\exists\Omega$ such that $\chi_i \in [\chi_j\rangle$ and $\chi_j \in[\chi_i\rangle$. This means that there is not any isolated nodes in the network and all nodes are interconnected. If $\exists\Omega$ such that $\chi_j\in \Omega$ and $\Omega \neq \phi$, then $\exists\chi_i$ such that $\Psi(\chi_i)$ is equal to $\Psi(\chi_j)$ and $\Psi(\chi_i)$ belongs to Θ.

5.3 Proof of the Soundness Theorem

To proof the "if and only if" soundness theorem, the proof is divided into two halves. The first half is to proof the theorem forward and the second half is to proof the theorem backwards. The first half of the proof states that if Ω is sound, then the three necessary conditions should apply. The conditions are:

Item 1: $\exists\Omega = X \times X$ such that $\Omega\neq \phi$.

Item 2: $\forall\chi_i, \chi_j, \chi_i \in\Omega, \chi_j\in\Omega, \exists\Omega$ such that $\chi_i \in [\chi_j\rangle$ and $\chi_j \in[\chi_i\rangle$, That implies that there are no isolated controllers and hence all controllers are connected.

Item 3: If $\exists\Omega$ such that $\chi_j\in \Omega$ and $\Omega \neq \phi$, then $\exists\chi_i$ such that $\Psi(\chi_i)$ is equal to $\Psi(\chi_j)$ and $\Psi(\chi_i) \in\Theta$.

Given that Ω is sound, therefore, any network component can send any other network component directly or indirectly.

$\therefore \chi_i \in [\chi_j\rangle$ and $\chi_j \in[\chi_i\rangle$
$\because \chi_i \in [\chi_j\rangle$ and $\chi_j \in[\chi_i\rangle$

$\exists\Omega$ such that $\Omega = X \times X$ and $\Omega\neq \phi$. For the SDN to be sound then if $\chi_i=\phi$, then $\exists\Theta$ such that $\Psi(\chi_i)= \Psi(\chi_j)$. If $\exists\Omega$ such that $c_j\in S$ and $S=\phi$, then $\exists\chi_i$ such that $\Psi(\chi_i)= \Psi(\chi_j)$ and $\Psi(\chi_i) \in\Theta$ is satisfied. $\therefore$ if Ω is sound, then the three conditions hold.

The second half of the proof seeks to proof the backward condition. That is if the three conditions are valid and satisfied then SDN is sound. Since $\exists\Omega=X \times X$ such that $\Omega \neq \phi$, therefore $\chi_i \in [\chi_j\rangle$, since $\chi_i \in [\chi_j\rangle$ and $\chi_j \in [\chi_i\rangle$ and since if $\exists\Omega$ such that $\chi_j \in \Omega$ and $\Omega = \phi$, then $\exists\chi_i$ such that $\Psi(\chi_i)= \Psi(\chi_j)$ and $\Psi(\chi_i) \in \Theta$. Therefore, SDN is always functioning and hence the SDN is considered to be sound.

At this point, the proof has been completed since the two halves of the proof were proven successfully. The mathematical prove discussed above is one of the important strengths of using petri-net in modelling various systems. $CPPN_{SDN}$ has the ability to validate conditions mathematically and prove it to be correct in both directions.

5.4 $CPPN_{SDN}$ Algorithm

This section discusses the $CPPN_{SDN}$ algorithm. $CPPN_{SDN}$ algorithm is explained in Algorithm 5.1 $CPPN_{SDN}$ uses the overall GPV to deploy the controller's placement. The $CPPN_{SDN}$ algorithm uses capacity, propagation delay, and packet loss to compute the GlobalPerformanceValue (GPV). Switches are assumed to be assigned to the controller that has the highest GPV.

Algorithm 5.1: Controller Placement using Petri-Nets for SDNs ($CPPN_{SDN}$)

1: Input topology $\Phi = (\upsilon, \varepsilon)$.
2: Compute capability matrix for all controllers based on their capacity, average propagation delay to their associated switches, and percentage of packet loss.
3: Select switch nodes with high average capability using petri-net mathematical model to form a new sub topology $\Phi' = (\upsilon', \varepsilon')$.
4: Calculate local density for each switch node i in Φ'.
5: Plot decision graph of Φ' to select sub-cluster centers where the center sets $V_{center} = \{\upsilon_1; \upsilon_2; \upsilon_k\}$.
6: Assign all switches to the nearest cluster center to the associated controller.
7: Each cluster center along with the switch nodes form a sub-network Φ_i.
8: The sub-network sets $\Phi_{sub} = \{\Phi_1; \Phi_2; \ldots; \Phi_k\}$
9: For each sub-network Φ_i, find the largest closeness centrality point as the new center. Update υ_{center}.
10: Reassign all switch nodes in Φ to the nearest cluster center, and update υ_{sub}.
11: Repeat steps 9 and 10 until each sub-network is no change.
12: Output result.

6 Experimental Results

This section has the experimental results of the work conducted using the $CPPN_{SDN}$. $CPPN_{SDN}$ is compared to the reference models. Internet2 OS3E topology is used during the evaluation process. The topology consists of 34 nodes and 42 links as shown in Fig. 6. The propagation delay among two switches is computed by dividing the geographical distance by 66.66% of the speed of light [20]. To verify the strength of the $CPPN_{SDN}$, $CPPN_{SDN}$ is compared to two reference models. The optimized K_{means} and the MDPC reference models. Fig. 6, Fig. 7, and Fig. 8 show the Internet2 OS3E network which is partitioned with k=2, 3, and 4 respectively by the $CPPN_{SDN}$. The arrows in the figures indicate the locations of the controllers. Fig. 9 and Fig. 10 have the average latency in the network on selected controllers for the $CPPN_{SDN}$ when compared to the optimized K_{means}.

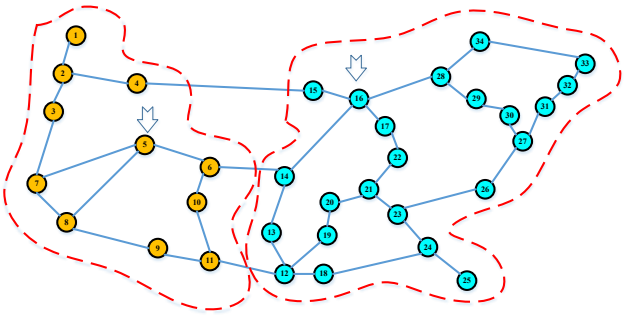

Fig. 6: Network Partition (k=2)

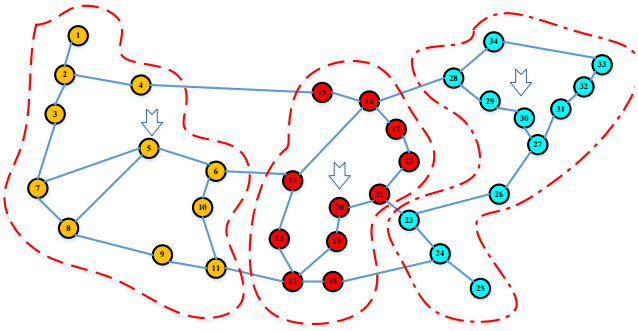

Fig. 7: Network Partition (k=3)

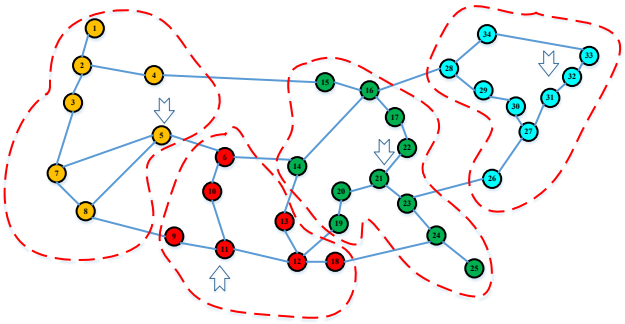

Fig. 8: Network Partition (k=4)

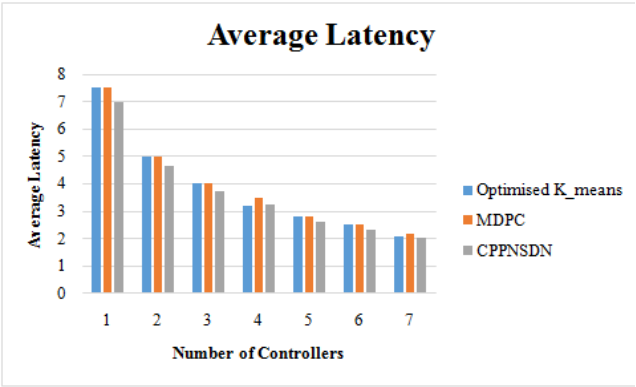

Fig. 9: Average Latency

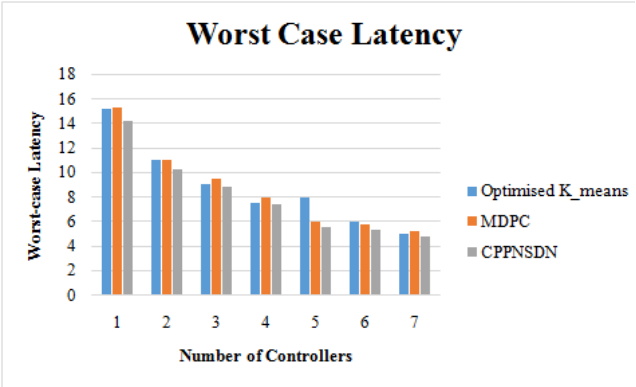

Fig. 10: Worst-case Latency

Results are very promising since $CPPN_{SDN}$ has shown significant improvement over two reference models. $CPPN_{SDN}$ has improved by 17% over the optimized K_{means} reference model and improved by 17% over the MDPC reference model. In the $CPPN_{SDN}$, the points with the maximum GPV are selected as the initialization to split the subnetwork into partitions. Because the $CPPN_{SDN}$ splits the maximum distance each time to subnet, the algorithm has good performance in terms of worst-case latency. The experiments were replicated three times with 90% confidence interval (CI). Standard deviation was computed as well.

7 Conclusion and Future Work

Controller placement problem (CPP) is an important problem when it comes to SDNs. CPP affects directly the overall networks' latency and performance. In this paper, we propose a new controller placement model that is based on the concept of petri-nets. Petri-nets strengths are in its generic modeling techniques. The proposed model is called Controller Placement using Petri-Nets for SDNs ($CPPN_{SDN}$). $CPPN_{SDN}$ models the controller placement problem to minimize the average propagation latency among switches and their associated controllers. $CPPN_{SDN}$ divides the network into sub-networks. Each sub-networks is governed through a controller. Experiments were conducted on the Internet2 OS3E topology to evaluate the performance of $CPPN_{SDN}$. Experimental results show that $CPPN_{SDN}$ reduces the average latency significantly. The average latency can be reduced when compared to reference models. Reference models used in this paper are Modified Density Peaks Clustering (MDPC) and Optimized K_{means}. In terms of the average overall latency, the $CPPN_{SDN}$ has shown very promising results as it outperformed the optimized mean by 7% and also $CPPN_{SDN}$ outperformed the MDPC reference model by 17%.

Experiments were repeated three times with confidence interval of 90%. Standard deviation was computed. This is an ongoing research, and different metrics could be used to measure the goodput of the proposed model such as the controller cost and the calculation cost. On the other hand, another direction of the future work is by adding other use cases such as failover scenarios, and comparing the results with the reference models.

References

[1] Jensen, K. (2013). "*Coloured Petri nets: basic concepts, analysis methods and practical use (Vol. 1)*. Springer Science & Business Media.

[2] B. Heller, R. Sherwood, and N. McKeown, "*The controller placement problem*," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 7–12

[3] S. Guo, S. Yang, Q. Li, and Y. Jiang, "*Towards controller placement for robust software-defined networks,*" in Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance. IEEE, 2015, pp. 1–8.

[4] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "*Dynamic controller provisioning in software defined networks*," in Network and Service Management (CNSM), 2013 9th International Conference on. IEEE, 2013, pp. 18–25.

[5] Y. Zhang, N. Beheshti, and M. Tatipamula, "*On resilience of split-architecture networks*," in Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE. IEEE, 2011, pp. 1–6.

[6] J. MacQueen et al., "*Some methods for classification and analysis of multivariate observations*," in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297

[7] A. Jalili, V. Ahmadi, M. Keshtgari, and M. Kazemi, "*Controller placement in software-defined wan using multi objective genetic algorithm*," in Knowledge Based Engineering and Innovation (KBEI), 2015 2nd International Conference on. IEEE, 2015, pp. 656–662.

[8] Wael Hosny Fouad Aly, Yehia Kotb, "*Towards SDN Fault Tolerance using Petri-Nets*," The 28th International Telecommunication Networks and Application Conference (ITNAC 2018), Sydney, Australia, 21-23 Nov, 2018.

[9] Wael Hosny Fouad Aly, "*LBFTFB Fault Tolerance Mechanism for Software Defined Networking*", The International Conference on Electrical and Computing Technologies and Applications, 2017 (ICECTA'2017), AURAK, UAE, Nov 2017.

[10] Wael Hosny Fouad Aly, "*A Novel Fault Tolerance Mechanism for Software Defined Networking*", European Modelling Symposium on Mathematical Modelling and Computer Simulation, Manchester, UK, Nov 2017.

[11] Wael Hosny Fouad Aly, Abeer Mohammad Ali Al-anazi, "*Enhanced Controller Fault Tolerant (ECFT) Model for Software Defined Networking*," The 5th IEEE International Conference on Software Defined Systems (SDS), Barcelona, Spain, April 2018.

[12] Kristensen, L. M., Christensen, S., & Jensen, K. (1998). "*The practitioner's guide to coloured Petri nets"*. International Journal on Software Tools for Technology Transfer (STTT), 2(2), 98-132.

[13] Chang, X., Pang, H., & Hu, L. (2010, August), "*Distributed computer network model base on petri nets"*. In Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on (Vol. 1, pp. 200-203)..

[14] Y. Zhang, S. Chen, and G. Yu, "*Efficient distributed density peaks for clustering large data sets in mapreduce*," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 12, pp. 3218–3230, 2016

[15] Singh, S., Singh, G., Narasimhan, V. L., & Pabla, H. S. (2014, January). "*Petri net modelling and analysis of mobile communication protocols UMTS, LTE, GPRS and MANET"*. In Computer Communication and Informatics (ICCCI), 2014 International Conference on (pp. 1-9).

[16] A. Rodriguez and A. Laio, "*Clustering by fast search and find of density peaks*," Science, vol. 344, no. 6191, pp. 1492–1496, 2014.

[17] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "*A k-means-based network partition algorithm for controller placement in software defined network*," in Communications (ICC), 2016 IEEE International Conference on. IEEE, 2016, pp. 1–6.

[18] Yuezhen Qi, Dongbin Wang, Wenbin Yao,Yuhua Cao, "*Towards Multi-Controller Placement for SDN Based on Density Peaks Clustering,*" ICC 2019 - 2019 IEEE International Conference on Communications (ICC) Shanghai, China, May 2019

[19] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/

[20] D. Klein and M. Jarschel, "*An openflow extension for the omnet++ inet framework*," in Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 322–329.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

Dr. Wael Hosny Fouad Aly carried out the following tasks:

- Modeling and simulation the CPPN.
- Implemented Algorithm 5.1.
- Organized and executed the experiments of Section 6 .
- The paper is not funded by any organization. It is my own work and efforts using simulation tools.