Sensors & Transducers, Vol. 195, Issue 12, December 2015, pp. 77-82

www.sensorsportal.com

Sensors & Transducers

http://www.sensorsportal.com

A Dynamic Deployment Policy of Slave Controllers for Software Defined Network

Yongqiang Yang and Gang Xu

College of Computer and Information Engineering, Henan University of Economics and Law, Zheng Zhou, 450002, China

E-mail: xgtony@163.com

Received: 5 November 2015 /Accepted: 30 November 2015 /Published: 30 December 2015

Abstract: In order to solve the problem of Software-Defined Network scalability, the idea of using multiple controllers to manage the underlying network domain became an effective solution. But how to deploy the controller rationally to reduce the communication overhead of control plane lacks the corresponding research. Therefore this paper proposes a controller dynamic deployment strategy. The strategy uses the Quantum Genetic Algorithm to calculate the optimal controller deployment scheme, and can adjust scheme of deployment dynamically according to the change of network topology. Experiments show that this deployment scheme can effectively reduce the whole communication overhead of the control plane, and achieve dynamic adjustment of deployment scheme at the smaller expense. *Copyright © 2015 IFSA Publishing, S. L.*

Keywords: Software-Defined Network, Controller, Dynamic deployment, Communication overhead.

1. Introduction

As the increasing of Internet scale and the growing emergence of new network services, the data forwarding plane and the tightly coupled mode in the routing control plane make the switch/router equipment control function of highly complex, which greatly increase the difficulty and the cost of network maintenance and management. Therefore, researchers put forward the idea of separating control logic and data forwarding to reduce the complexity of the network core equipment. This can also improve the network management and control of flexibility. Based on the idea, the Clean slate team in the Stanford University proposed the Software-Defined Network (SDN) [1]. The appearance of SDN gives revolutionary influence on the new network system research. SDN has become the one of the most popular research direction in educational circles and industry.

In SDN, all switches at the bottom of the data forwarding rules are generated by controllers, and issued through the OpenFlow [2] to each switch. The controller is a kind of centralized management node. If the entire network uses a single controller, it is difficult to secure the scalability and reliability of the network. If concurrent traffic has increased dramatically at a certain moment, a single controller may not be able to respond to a large number of concurrent requests of OpenFlow switch. The OpenFlow switch cannot arrive on a large number of packet forwarding, which lowered the overall performance of network. And the switch will not be able to cope with the new network flow when the single controller fails. Therefore, using multiple controllers for distributed management becomes the development direction of SDN. The number and position of the controller directly determines the performance and efficiency of the network deployment.

Some scholars put forward more than two kinds of controller respectively management framework HyperFlow [3] and Kandoo [4] in the study on multiple controller at present. Literature [5] proposed a management node deployment scheme for Pressure. The architecture put forward a set of centralized network state information collection, aggregation, and filtering strategy. The Pressure deployed strategy by calculating Pressure score to develop information aggregation points. The HotSpot proposed in literature [6] formulated the management node deployment strategy by calculating the number of neighbor nodes for each node. The Pressure and HotSpot cannot automatically obtain the optimal quantity of node value, which can only rely on the administrator to set in advance. The two methods defined the distance between the nodes as the hop count between nodes simply without considering the effects of parameters such as delay.

The intelligent optimization algorithm simulates some natural phenomenon and the process to solve complex optimization problems. Genetic algorithm is a kind of intelligent optimization algorithm, which has a wide range application. It simulates the biological genetic and evolutionary process to solve discrete optimization problems. Narayanan [7] combined quantum computation theory and genetic algorithm for the first time, and proposed the Quantum Genetic Algorithm. Han [8] further introduced the quantum revolving door into the quantum genetic algorithm, and also appeared some improvements of the quantum revolving door [9].

This paper mainly studied the controller deployment problem of SDN. We describe a two level control system of SDN, and then proposed a controller deployment strategy based on quantum genetic algorithm. In addition, we design a dynamic adjustment strategy of controller to guarantee the stability of the system. Finally the experiments verify that the control strategy can effectively reduce the SDN flat overall communication overhead, and can adapt to the change of the underlying network topology by the small dynamic adjustment.

2. The Slave Controller Based on Quantum Genetic Algorithm Deployment Strategy

The Slave controller deployment plans will have a big impact on the efficiency of the whole control plane. Too much Slave controller will make the Slave controller and the interaction between the Master controllers of communication resource depletion. Too little Slave controller will also increase response delay in nodes at the bottom of the Slave controller. The best deployment strategy of Slave controller can control the plane with the minimum overall communication overhead to realize the control for multiple administrative domains. So the Slave controller deployment is converted into a 0-1 programming problem. The problem model is as follows:

Variables:

$\boldsymbol{x_i}$: Binary Variables.

If the Slave controller is connected to the i^{th}-switch, the x_i is 1, otherwise the x_i is 0.

Input:

$\boldsymbol{S}$: The underlying node collection.

$\boldsymbol{M}$: The short circuit matrix.

$\boldsymbol{M}(s, t)$ is communication overhead for the source s to the destination endpoint t.

$\boldsymbol{M}$: The serial number of the Master controller.

The objective function:

$$\min \sum_{i=1}^{|S|} x_i \cdot M(i,m) + \sum_{i=1}^{|S|} (1-x_i) \cdot M(i,c_i), \quad (1)$$

which defines the control plane of the whole communication overhead.

Constraints:

$$c_i = \{n \mid n \in S, x_n = 1, M(i,n) = \arg\min_{j \in S, x_j = 1} M(i,j)\}, \; x_i \in \{0, 1\}, \quad (2)$$

where c_i is responsible for the control of the i^{th} node at the bottom of the Slave controller.

In order to make the network control consumption overall communication overhead lowest, the Slave controller deployment plans need to determine how many Slave controller is set in the underlying network, and determine what the underlying node is set to the communication agent. Because of the good performance of the Quantum Genetic Algorithm in dealing with discrete optimization problems, this paper proposes a Slave Controllers Deployment strategy based on the Quantum Genetic Algorithm (SCD - QGA). The strategy using the Quantum Genetic Algorithm solves the 0-1 programming problem, and draws the optimal Slave controllers deployment scheme.

2.1. The Quantum Genetic Algorithm

The Quantum genetic algorithm is based on the concept of quantum bit and quantum superposition. The quantum bit is the smallest unit of information in quantum computation. The quantum bits can be in a state | 0 >, | 1 > state, and arbitrary superposition between | 0 > and | 1 >. Its state can be described as:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (3)$$

where α, β are the plural and denote respectively quantum bit for | 0 > and | 1 > probability amplitude. $|\alpha|^2$ denote that the quantum bit is observed as | 0 > state probability. $|\beta|^2$ denote that the quantum state is observed for | | 1 > state probability. They satisfy the normalization conditions:

$$|\alpha|^2+|\beta|^2=1 \quad (4)$$

The operand of Quantum Genetic Algorithm is composed of multiple feasible solutions. The feasible solution can be considered as an individual chromosome. Each chromosome is composed of multiple quantum bit. The probability of a quantum bit can be defined as $[\alpha\ \beta]^T$, and a m quantum bit of chromosome encoding for:

$$q=\begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_m\\ \beta_1 & \beta_2 & \cdots & \beta_m\end{bmatrix}, \quad (5)$$

Each genes in the chromosome can be | 0 > state, 1 > | state and superposition of | 0 > and | 1 >. A chromosome can characterize the state of 2^m. After determining the initial chromosome coding, the measurement of chromosome can calculate the fitness of individuals in the calculate population. Measure is to make the chromosomes each quantum bit collapse into a certain state. Each quantum bit generates a random number. If the random number is less than $|\alpha|^2$, the quantum bits of measure-ment value is 0, or 1. It can be done in quantum revolving door populations of updates in the iteration process of evolution. The Quantum revolving door is a unitary matrix, which is used to alter the probability amplitude of quantum superposition.

2.2. The Generation Process of Deployment Scheme

The Quantum Genetic Algorithm is used to calculate the Slave controller deployment scheme in this paper. First we need to calculate the input parameter *M* to solve the cost minimum of each node to all other nodes at the bottom of the communication. So the problem can be transformed into solving the shortest path in the network.

Step 1: First, the underlying network topology is converted into a weighted undirected graph. The weight of each edge is the delay of the link at the bottom.

Step 2: Then we call the Bellman-Ford algorithm [10] repeatedly and calculate the monophyletic shortest path from each of the underlying nodes to other nodes in turn. The numbers are recorded in the matrix *M*.

Step 3: The *m* deployment scheme are generated randomly and coded to form the initial population. Then the Quantum Genetic Algorithm is called to calculate. When evolution we need to compute each individual's adaptive value Fit(*X*). Before calculating the adaptive value, each of the node at the bottom will be specified a fixed Slave controller c_i. In the SCD - QGA strategy the c_i is set for the Slave controller with minimum communication overhead with node *i*. The adaptive value mainly depends on the size of the whole communication overhead.

$$\mathrm{Fit}(\boldsymbol{X})=\left[\sum_{i=1}^{|S|}x_i\cdot M(i,m)+\sum_{i=1}^{|S|}(1-x_i)\cdot M(i,c_i)\right]^{-1} \quad (6)$$

The algorithm is terminated after the evolution for *N* round. The output of the optimal solution in the process of evolution is considered as the final deployment scheme, which deploy the Slave controllers according to the scheme. The strategy implementation process is as follows:

Input: the population individual's number *m*, the maximum evolution algebra *N*, the chromo-some's length *s*, the mutation probability *p*
Output: the optimal scheme of $\boldsymbol{X}_b$

1 Call the Bellman - Ford algorithm and calculate the short circuit matrix $\boldsymbol{M}$.
2 $t\leftarrow 0$, Generate original population $Q(t)$ and its individual number is *m*. Initialize the chromosome.
3 The $Q(t)$ is measured and the measurement value is generated as $P(t)$.
4 **for all** $\boldsymbol{X}_i \in P(t)$ **do**
5 Query the matrix $\boldsymbol{M}$. Specify a minimum communication cost of Slave controller for each switch.
6 Calculate Fit($\boldsymbol{X}_i$).
7 **end for**
8 $\boldsymbol{X}_b=\{\boldsymbol{X}_i\,|\mathrm{Fit}(\boldsymbol{X}_i)=\arg\max_{X\in P(t)}\mathrm{Fit}(\boldsymbol{X})\}$;
9 **while**($t<N$)
10 $Q(t)$ evolve to $Q(t+1)$ with the quantum revolving door. $t=t+1$.
11 The $Q(t)$ is measured and the measurement value is generated as $P(t)$.
12 **for all** $\boldsymbol{X}_i \in P(t)$ **do**
13 Query the matrix $\boldsymbol{M}$. Specify a minimum communication cost of Slave controller for each switch.
14 Calculate Fit($\boldsymbol{X}_i$).
15 **end for**
16 $B(t)=\{\boldsymbol{X}_i\,|\mathrm{Fit}(\boldsymbol{X}_i)=\arg\max_{x\in P(t)}\mathrm{Fit}(\boldsymbol{X})\}$;
17 **if** Fit($\boldsymbol{X}_b$)< Fit($B(t)$) **then**
18 $\boldsymbol{X}_b = B(t)$;
19 **end while**
20 Output $\boldsymbol{X}_b$

3. The Adjust Strategy of Dynamic Slave Controller

The underlying network is dynamic change in the process of SDN operation. The underlying node or link failure, a new node to join the network and so on will cause the change of the underlying network topology. The original deployment plan of Slave controller is no longer the optimal solution. From the perspective of reduce the communication cost control plane, this section presents a deployment solution strategy with dynamic adjustment Slave controller.

The deployment scheme is recalculated by using SCD - QGA for the changed network topology, and the deployment of the Slave controller is adjusted according to the new plan. There may be a huge difference between the calculated new scheme with the original plan. Most of the Slave controller needs to be deployed. This will greatly increase the adjustment cost, and also can bring negative effect to the stability of the control plane. So this method can hardly be accepted by the operators of SDN. It is a reasonable way of adjustment that the Slave controller adjustment range is limited in reassigning domain, and thus only to adjust the deployment location of the individual Slave controller.

This paper proposes a Dynamic Slave Controllers Reassigning policy of Slave controller. The strategy will evaluate the effects caused by the change of topology to control plane, and adjust range constraint in the affected areas. Finally adjustment scheme of the Slave controller is calculated by the Quantum Genetic Algorithm.

3.1. The Setting of Adjust Domain

The change of network topology can be divided into four types: node failure, link failures, the new node and the new link. The link failure and new link specifically is the link changes when a node has not changed. Table 1 gives the four types of topology and the corresponding adjustment domain set strategy.

Table 1. The adjustment domain set strategy.

The type of topology change	The adjustment domain set strategy
Node failure	The neighbor node of failure node in the management domain
Link failures	Invalid link node in the management domain
New node	The neighbor node of the new node in the management domain
New link	The new link node in the management domain

3.2. The Process of Dynamic Adjustment

After confirmed the adjustment domain, the DSCR will call Quantum Genetic Algorithm to calculate the new Slave controller deployment scheme. During initialization, The Slave controller adjustment range is limited to domain. The chromosome of quantum bits digit is set to the number of nodes in the adjustment domain. Each quantum bit on behalf of a node in the adjustment domain. In order to accelerate the convergence speed, initialization of generation 0 chromosome, each value of α and β will be set in accordance with the Slave controller of in the adjustment domain and the proportion of the bottom node.

The DSCR measure pairs of chromosomes firstly in the fitness calculation and generate the Slave controller deployment plan in the adjustment domain. Then adjust the plan with the existing deployment in the outside to merge the deployment of the strategy, so as to get a new deployment plan. The type (6) can be used to calculate the adaptive value of the new scheme.

There are certain differences in the initialization and the calculating of the adaptive value. The other evolutionary step of DSCR and the whole evolution process are the same as the SCD - QGA. After evolution for N round, the optimal solutions of output in the process of evolution is considered as a new deployment plan, and adjust the domain to redeploy the Slave controller according to the scheme. All the nodes in the domain will divide the administrative domain according to the communication overhead of the Slave controller.

4. The Analysis of Simulation Results

This experiment runs in the PC of Pentium 4, 3.2 GHz CPU and 1 Gb of memory. The underlying network topology is generated by the tool of GT - ITM [11]. The underlying link delay value distribute within [1,100] uniformly. The simulation is implemented in Matlab. The simulation results of SCD - QGA are compared with the results generated by the Pressure [5], the HotSpot [6] Slave controller deployment. The results of simulation will also be compared with the deployment strategy based on the Genetic Algorithm. The population size of SCD - QGA is set to 20.

First of all, the number of Slave controller calculated for each deployment strategy is analyzed. The Pressure and HotSpot can not determine the optimal number of Slave controller automatically, which can only rely on specify beforehand of manager. The number of Slave controller Settings will have great influence on communication overhead.

The deployment strategy based on intelligent algorithm can automatically calculate the optimal number of Slave controller and the deployment scheme calculated is not affected by artificial Settings.

If the proportion of total Slave controller node set to 11 % and 18 % respectively and the size of the underlying network is 100, the Fig. 1 shows that the scheme calculated from the Pressure and HotSpot are the best. When the size of the underlying network is 500, the proportion of the Slave controller node set to 10 % to get the optimal solution of the above two strategies.

The proportion of the Slave controller is not fixed in the deployment scheme calculated by the SCD – QGA. Different proportion of the underlying network have different Slave controller (as shown in Fig. 2). The scope of the value is between [0.05, 0.15] generally.

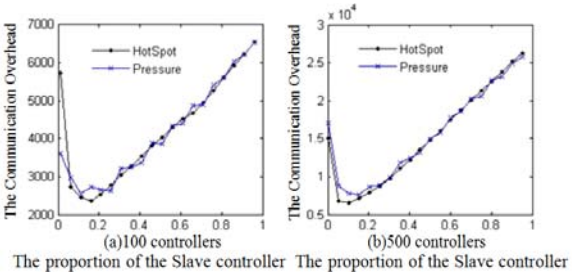

Fig. 1. The number of Slave controller for the influence of Pressure and HotSpot.

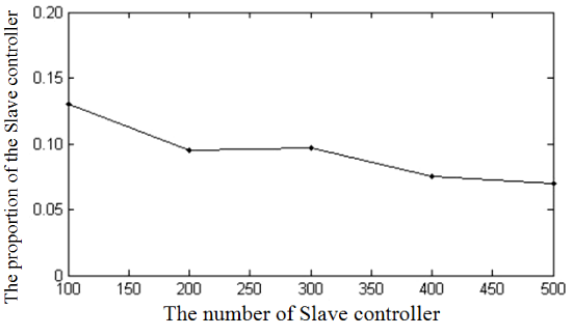

Fig. 2. The proportion of Slave controller of SCD-QGA under different network size.

The evolution algebra n is the number of iterations of the Genetic Algorithm. The solution for the algorithm is better if the value of n is greater. The optimal solution for the HotSpot and Pressure is only affected by the percentage of the Slave controller. Thus the deployment scheme of the HotSpot and Pressure is regarded as the two strategies under the optimal proportion (18 % and 11 %) generated by the scheme. If the underlying network scale is 100, Fig. 3 shows the evolutionary process of the various strategies generating the optimal deployment of schemes. It can be seen that the convergence rate of the SCD - QGA is the fastest and the deployment scheme of communication overhead is minimal, which is followed by GA. The deployment scheme calculated by the two strategies is better than the HotSpot and Pressure.

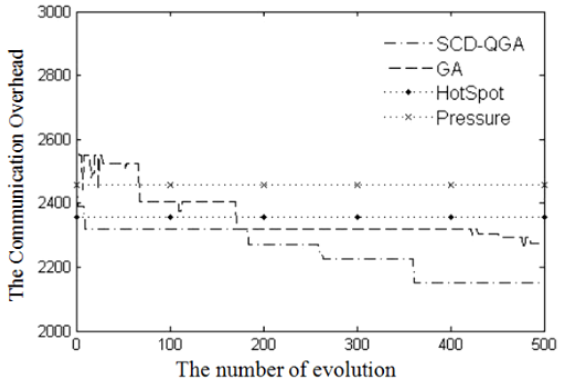

Fig. 3. The evolutionary process of the optimal solution.

In order to test the influence of the dynamic change of network topology to the communication overhead for control plan, the experiment generates 100 network topology changed randomly. We test the initial deployment scheme generated by the SCD - QGA under the new topology, the recalculate deployment scheme reSCD - QGA generated by the SCD - QGA and the deployment scheme generated by the DSCR. The initial network topology contains 100 nodes. The biggest evolution algebra of quantum genetic algorithm is set to 500. By shown in Fig. 4, the scheme of reSCD - QGA and DSCR in terms of communication overhead is lower than static initial deployment scheme. The difference between the new topology and the initial topology is becoming bigger and bigger especially with the constant changing of topology. The advantages of reSCD - QGA and DSCR are becoming more and more obvious.

The biggest advantage of DSCR scheme comparing with reSCD - QGA is adjusting less Slave controller to get better effect. The number of changed deployment Slave controller is called adjustment scale in the process of the adjustment of the Slave controller. By as shown in Fig. 5, the reSCD - QGA adjustment scale to an average is 6.65 in the process of network topology changes, while the DSCR adjustment scale to an average is 1.72. It can be seen that the DSCR adjustment scale significantly is less than the reSCD - QGA scale. Therefore using DSCR deployment scheme of dynamic adjustment can ensure the stability of the control plane.

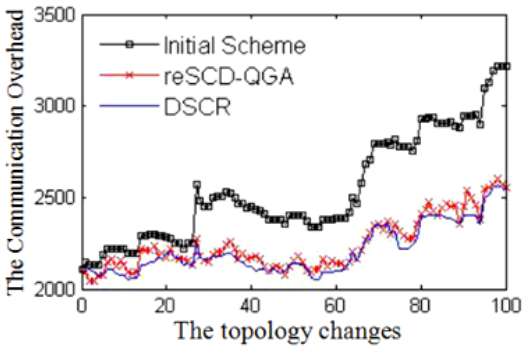

Fig. 4. The communication overhead of dynamic network topology changes.

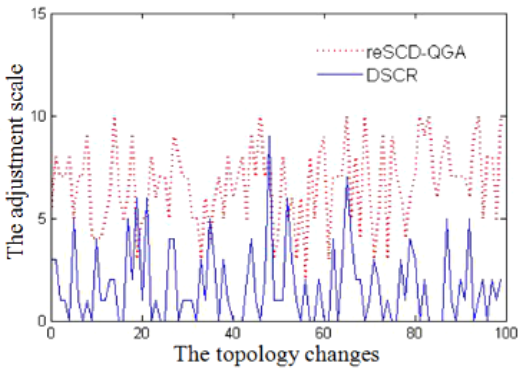

Fig. 5. The adjusting size of the Slave controllers.

5. Conclusion

In this paper, we design a dynamic deployment strategy of the SDN controller. The scheme reduces the communication overhead of control plane by setting the Slave controller. To minimize the communication overhead of the whole control plane, the scheme firstly transforms the Slave controller deployment issues to 0-1 programming problem and solves the problem by using the Quantum Genetic Algorithm to get optimal deployment plan. In addition, we design a Slave controller adjust strategy to reduce the effect caused by the change of dynamic topology on the stability of the control plane. The simulation experiments show that the proposed strategy not only the administrator need not set manually the number of Slave controller, but also the control plane consumed needs less communication overhead comparing the generated deployment scheme with other scheme.

References

[1]. Open Network Foundation (ONF). Software defined networking: the new norm for networks [EB/OL], https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn- newnorm.pdf

[2]. Mckeown N., Anderson T., Balakrishnan H., et al., Open-Flow: enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review,* 38, 2, 2008, pp. 69- 75.

[3]. Tootoonchian A., Ganjali Y., HyperFlow: a distributed control plane for OpenFlow, in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking (INM/WREN'10)*, 2010, pp. 3-3.

[4]. S. Hassas Yeganeh, Y. Ganjali. Kandoo, A framework for efficient and scalable offloading of control applications, in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN'12),* 2012, pp. 19–24.

[5]. R. G. Clegg, S. Clayman, G. Pavlou, L. Mamatas, On the selection of management/monitoring nodes in highly dynamic networks, *IEEE Trans. Computers,* 62, 6, 2013, pp. 1207-1220.

[6]. Mamatas L., Clayman S., Charalambides M., et al., Towards an information management overlay for emerging networks, in *Proceedings of the Network Operations and Management Symposium*, Osaka, Japan, 2010, pp. 527-534.

[7]. Narayanan A., An introductory tutorial to quantum computing, in *Proceedings of the IEEE Colloquium on quantum Computing Theory, Applications and Implications,* London, 1997, pp. 1-3.

[8]. Han K. H., Park K. H., Lee C. H., et al., Parallel quantum inspired genetic algorithm for combinatorial optimization problem, in *Proceedings of the 2001 Congress on Evolutionary Computation,* USA, 2001, pp. 1422-1429.

[9]. Xing Huan-Lai, Pan Wei, Zou Xi-Hua, A novel improved quantum genetic algorithm for combinatorial optimization problems, *Acta Electronica Sinica,* 35, 10, 2007, pp. 1999-2002.

[10]. Bellman R. Dynamic Programming, *Princeton University Press,* Princeton, 1957, pp. 124-126.

[11]. Zegura E, Calvert K, Bhattacharjee S., How to model an internetwork, in *Proceedings of the IEEE International Conference on Computer Communications*, San Francisco, USA, 1996, pp. 594 – 602.