Load Balancing Strategy of SDN Controller Based on Genetic Algorithm

Seung-Bo Kang, Gu-In Kwon

Department of Computer Science and Engineering,
Inha University, 100 Inha-ro, Nam-gu Incheon 22212, KOREA
gogetit88@gmail.com, gikwon@inha.ac.kr

Abstract. Software Defined Network (SDN) provides centralized and opened way to manage the network by separating both control plane and data plane each other. The controller defined as a software is responsible for a control plane. If there is too much load on a specific controller, controller’s suboptimal performance problem will be occurred. Because of this problem, network will provide services in an inefficient way. OpenFlow provides a way to connect switches with multiple controllers so that the controllers’ load can be balanced. This paper suggests a solution to distribute and balance the load on controllers based on genetic algorithm when imbalance is detected. Then we can get an optimized way to deploy the switches to controller and network services can be provided efficiently.

Keywords: SDN, OpenFlow, genetic algorithm, load balancing, reliability

1 Introduction

Being a closed and distributed character of the legacy network, it is difficult to manage a network. And it can’t be flexible and scalable. To deal with it, Software Defined Network (SDN) and OpenFlow protocol which provides open environment is proposed as a new paradigm.

Current SDN controllers provide [1] [2] many service operations such as routing topology processing, firewall, VPN and more. When the number of the switches connected to a particular controller is concentrated and the amount of load to the controller is overloaded, suboptimal performance problems can be occurred. Then the situation that controllers cannot take the requests from switches will occur. OpenFlow supports that switches can have connections with multiple controllers to solve this problem [3]. This paper suggests a solution to distribute and balance the load of controllers based on genetic algorithm when imbalance is detected.
2 Load balancing strategy of SDN controller based on genetic algorithm

2.1 Population coding

We initiate a population to generate mapping solutions which are the ways to assign switches to controllers randomly. Mapping solutions belonging to the population can produce a new good mapping solution where loads are well distributed to each controllers via the selection, crossover and mutation operations according to the genetic algorithm.

Also, because the controllers and switches are one-to-many relation, we represent a mapping solution as a tree structure. A super controller responsible for load balancing operation is a root node, and a controller is an internal node and a switch is a leaf node.

2.2 Fitness function

We use a fitness function \(f(M_i, T)\) from an existing study of the cloud computing [4] to obtain a fitness value of mapping solution \(M_i\) for a time \(T\). The fitness function determines the suitability of the mapping. If the mapping solution has high fitness value, the probability to pass the genes to next generation will be also high.

\[
f(M_i, T) = \frac{1}{A + B \times f_H} \tag{6}
\]

\[
f_H = \Phi(\sigma_i(M_i, T) - \sigma_0), \quad \Phi(X) = \begin{cases} 1, & X \leq 0 \\ r(>1), & X > 0 \end{cases} \tag{7}
\]

\(\sigma_0\) is a predefined standard deviation of load and \(f_H\) is a difference value between \(\sigma_0\) and a standard deviation of a current mapping solution. A and B is a weighed value to tune the sensitivity of standard-deviation difference in the fitness function. After the calculation of all individual’s fitness value, selection operation will be done as a next procedure.

2.3 Selection operation

A roulette-wheel selection method is used to select parent chromosome. This selection method is the way to select mapping solutions according to the ratio of the fitness value of each individual to the fitness value of all individual [5]. This paper calculates the probability as follows.

\[
p(M_i) = \frac{f(M_i, T)}{\sum_{i=1}^{N} f(M_i, T)} \tag{8}
\]

\(p(M_i)\) means \(M_i\)’s probability to be selected. After the selection, new individual will be generated through crossover operation.
2.4 Crossover operation

We can find better mapping solution whose loads are spread well using crossover operation which recombines each individual’s genes. The following steps show how to perform crossover operation to make a new mapping solution \( M_0 \).

1. Select the mapping solutions \( M_1, M_2 \) after calculation of each individual’s fitness value.
2. Assign any switch which is duplicated on each mapping solution \( M_1, M_2 \) to a new solution \( M_0 \).
3. Assign remaining switches after step 2 to the new solution \( M_0 \).
   Each switch computes its load and we rank the switch based on their load. A switch which has the largest load will be assigned to the least loaded controller and the switch having the next largest load is assigned to the next least loaded controller.
   This assignment will continue until all switches are assigned to the controllers.
4. If the standard deviation of a new mapping solution’s load is lower than predefined value, we can use the mapping solution. If it isn’t, go back to step 2 and do this algorithm again.

2.5 Mutation operation

Genetic algorithm uses a mutation operation to vary the operational changes preventing overly enthusiastic or potential loss of valuable genes after crossover operation [6]. Proposed algorithm in this paper does mutation operation by selecting the individual randomly according to predefined probability \( P_m \) on selection operation. It can widen the spectrum of result.

3 Conclusions

In this paper, by using the genetic algorithm we proposed a meta-heuristic way to balance the load. If a load given to a controller beyond the pre-given threshold, our load balancing algorithm distributes the load. With this algorithm, as well as to distribute the load, an optimized mapping solution under the given conditions can be obtained.

Acknowledgement. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (no. 2015R1A2A2A01003501).

References

3. OpenFlow switch specification 1.5.1, https://www.opennetworking.org/