Virtual SDN Network Embedding Algorithm Based on Load Balance

Jinpeng Ran$^a$ and Xiang Wang$^b$

Information and Navigation College, Air Force Engineering University, Xian, China. \(^a^\)ranjp120@163.com; \(^b^\)wangxiang_626@hotmail.com

Abstract. Aiming at Software Defined Network (SDN) virtualization environment, a virtual SDN (vSDN) network embedding method based on fruit fly algorithm is proposed to minimize the overall load imbalance and average control delay of the network. The simulation indicates that the proposed method improves the vSDN request acceptance rate. Besides, it has good load balance performance of the underlying physical SDN network, and reduces the communication delay for controller-to-switch.

1. Introduction

The emergence of Software Defined Network (SDN) and Network Virtualization (NV) has just brought about an opportunity to solve the ossification problem of traditional networks. SDN separates the data plane from the control plane, and uses a logical centralized controller to uniformly manage data distribution and conversion devices, so that each physical device performing the task can independently complete the corresponding functions. NV extracts, isolates, and reconstructs the underlying physical network entity resources, and builds multiple virtual networks on a common physical network, thereby the underlying network resources are shared. The idea of SDN centralized control just provides a flexible and open new way to realize NV. Therefore, NV technology based on SDN has become a hot and meaningful issue [1].

As one of the key technologies of NV, virtual network embedding refers to the abstraction of diversified services into virtual network, and map to specific nodes and links in the underlying physical network according to their own resource requirements. In the virtual network environment based on SDN, the extracted virtual SDN network is called vSDN. Each vSDN is managed and controlled by a virtual Controller (vC) deployed at the virtual network node. In order to meet the communication requirements of the vSDN request, the controller-to-switch in the embedded physical SDN should have a small communication delay.

At present, researches on virtual network embedding methods based on SDN architecture are gradually increasing. Capelle et al. focused on the network forwarding resources, and designed an integer linear programming model to solve the online resource allocation problem of multiple virtual links in the SDN environment in Ref. [2]. Aiming at the resource embedding problem in virtual data center, Rosa et al. proposed an adaptive embedding algorithm of bandwidth resource guarantee, which realized load balance and improved resource utilization in Ref. [3]. However, the communication delay problem of controller is rarely involved in the most research. In this paper, we proposed a novel method named vSDN embedding based on fruit fly optimization algorithm (FOA-vSDNE) which optimized the load balance performance of the underlying network on the basis of considering the deployment on vSDN controller. This paper is organized as follows: Section 1. briefly introduces the background of SDN and NV. Section 2. shows the model analysis for virtual SDN network embedding.
problem and algorithm design. Section 3. conducts simulation experiments and analysis of results. Finally, Section 4. concludes the paper and points out further work.

2. Theoretical Analysis

2.1. Model Analysis for vSDN Network Embedding Problem

The underlying physical SDN network is denoted as a weighted undirected graph $G = (N, L, C^N, C^L)$, where $N$ and $L$ represent the sets of network nodes and links. $C^N$ is the attribute set of physical node $n_i$ ($n_i \in N$), including the available computing resource CPU ($n_i$) of the node, and the flow table storage processing resource FTSP ($n_i$) in the switch. $C^L$ is the attribute set of physical link $l_i$ ($l_i \in L$), including link available bandwidth resource $b(l_i)$ and delay capability $d(l_i)$. We assume that $P_s$ is the set of all acyclic paths interconnected by nodes in the underlying physical SDN network, and any path $p_s$ satisfies $p_s \in P_s$. The available bandwidth of the underlying physical link is defined as the difference between the total bandwidth of the link and the vSDN request bandwidth embedded to the link, and the available bandwidth $b(p_s)$ on $p_s$ is defined as the sum of the delays of each link on the path.

Similarly, a vSDN network is also represented by $G = (N, L, C^N, C^L), n_i, n_j \in N$, where $N$ and $L$ represent the sets of virtual network nodes and links. Virtual node $n_i$ and virtual control node $n_c$ are included in $N_i$. The virtual control node is deployed at the location of the underlying physical node, which does not occupy the node switch resources, and belongs to a special type of virtual node. Virtual link $l_i$ and virtual control link $l_c$ are included in $L$. The virtual control link set is represented by $L_c$ ($L \subseteq L_c$). $C^N$ is the resource constraint set of virtual node $n_i$, including the CPU resource CPU($n_i$) and FTSP resource FTSP($n_i$) requested by the virtual node. $C^L$ is the attribute set of virtual link $l_i$, including virtual link bandwidth requirement $b(l_i)$.

The vSDN network embedding includes two stages of virtual node embedding and virtual link embedding. Node embedding maps virtual nodes and virtual control nodes to physical nodes that satisfy CPU and FTSP resource requirements, and virtual control nodes are treated as virtual nodes with no resource requirements. The binary variable $X_s(i, j)$ is used to represent the node embedding relationship. When the virtual node $n_i \rightarrow n_j$, $X_s(i, j)=1$, otherwise 0. When the virtual control node $n_c \rightarrow n_i$, $X_s(c, j)=1$, otherwise 0. Link embedding maps virtual links and virtual control links to physical paths that satisfy bandwidth resource requirements. The binary variable $M_L(l_i, l_j)$ is used to represent the link embedding relationship. When the virtual link $l_i \rightarrow l_j$, $M_L(l_i, l_j)=1$, otherwise 0. When the virtual control link $l_c \rightarrow l_i$, $M_L(l_i, l_j)=1$, otherwise 0.

In this paper, the communication delay of controller and the network load imbalance are used as parameters to establish a multi-objective optimization network embedding problem model. Firstly, the network load imbalance is measured by the node and link load rate. The load rate is defined as the ratio of the total resource requests embedded to the node (link) and the original resource of the node (link). The relevant expressions are described as follows:

$$N(n_i^{m}) = \frac{\sum_{\forall n_i \in N} (CPU(n_i) + FTSP(n_i))}{CPU(n_i^{m}) + OFTSP(n_i^{m})}$$  \hspace{1cm} (1)

$$N^{m} = \frac{\sum_{m=1}^{\mid M \mid} N(n_i^{m})}{\mid M \mid}$$  \hspace{1cm} (2)

$$N^{m} = \sum_{m=1}^{\mid M \mid} (N(n_i^{m}) - N(n_i^{m}))^{2} / \mid M \mid$$  \hspace{1cm} (3)

Finally, in order to ensure the correctness of the set of solutions obtained, the model analysis for vSDN network embedding problem is performed.
\[ L(l'_s) = \sum_{\forall l \in L_s} b(l_s) / ob(l'_s) \]  \hspace{1cm} (4)

\[ \overline{L(l'_s)} = \sum_{\forall l \in L_s} L(l'_s) / |E| \]  \hspace{1cm} (5)

\[ L_{\alpha'} = \sum_{\forall l \in L_s} (N(l'_s) - L(l'_s))^2 / |E| \]  \hspace{1cm} (6)

\[ \text{Min } f_1 = \eta N_{\alpha'} + (1-\eta)L_{\alpha'} \]  \hspace{1cm} (7)

Where Eq. (1) represents single node load rate when a physical node is acting as a working node. Eq. (2) represents the average load rate of the \(|M|\) underlying physical SDN available nodes. Eq. (3) represents the overall load imbalance of the underlying network node. Eqs. (4-6) have a similar meaning. Eq. (7) represents the overall load imbalance of the underlying network, where \(\eta \in [0,1]\) is the adaptive adjustment parameter.

Secondly, the minimum average control delay formula is given:

\[ \text{Min } f_2 = \sum_{l_{Oh}=L_s} \sum_{l_{Oh}=L_s} M_L(cp,ij) d(l_{Oh}) / L_s \]  \hspace{1cm} (8)

Where \(|L_s|\) is the number of virtual control links in the vSDN request.

Combined with the vSDN embedding problem characteristic, the resource constraints of the node can be written as:

\[ \forall n_i \in N_v, \forall n_j \in N_v, \begin{cases} X_N(i,j) \cdot \text{CPU}(n_i) \leq \text{CPU}(n_j) \\ X_N(i,j) \cdot \text{FTSP}(n_i) \leq \text{FTSP}(n_j) \end{cases} \]  \hspace{1cm} (9)

\[ \begin{cases} \forall n_i \in N_v, \sum_{n_j \in N_v} X_N(i,j) = 1 \\ \forall n_j \in N_v, \sum_{n_i \in N_v} X_N(i,j) = 1 \\ \forall n_j \in N_v, \sum_{n_i \in N_v} X_N(i,j) \leq 1 \end{cases} \]  \hspace{1cm} (10)

Where Eq. (9) represents that the CPU resources and FTSP resources requested by the virtual node cannot exceed the CPU resources and FTSP resources of the underlying embedded physical node. Eq. (10) represents that for any virtual node, one virtual node in the same vSDN request can only be embedded to one underlying physical node. Eq. (11) represents that for any physical node, one underlying physical node carries at most one virtual node in the same vSDN request.

The resource constraints of the link can be expressed as:

\[ \forall l_{Oh} \in L_s, \sum_{l_{Oh}=L_s} M_L(\text{uh},ij) \cdot b(l_{Oh}) + \sum_{l_{Oh}=L_s} N_L(cp,ij) \cdot b(l_{Oh}) \leq b(l_{Oh}) \]  \hspace{1cm} (12)

\[ \forall n_i \in N_v, \forall l_{Oh} \in L_s, \sum_{l_{Oh}=L_s} M_L(\text{uh},ij) - \sum_{l_{Oh}=L_s} M_L(\text{uh},ji) = \begin{cases} 1, X_N(\text{uh},i) = 1 \\ -1, X_N(\text{uh},i) = 1 \\ 0, \text{otherwise} \end{cases} \]  \hspace{1cm} (13)

Where Eq. (12) represents that the sum of the bandwidth requirements of the link cannot exceed the available bandwidth resources of the underlying embedded physical link. Eq. (13) takes embedding virtual link \(l_{Oh}\) as an example, which indicates that after embedding, the virtual link \(l_{Oh}\) is embedded to a SDN network path \(p_i\) between the underlying physical nodes \(i\) and \(j\), and the link connectivity constraints should be met. \(l_{Oh}\) is similar to \(l_{Oh}\) in form and will not be listed.
2.2. Algorithm Design

For the bat position, the position vector of each fruit fly in the algorithm represents a solution of the vSDN network embedding problem, which is a possible embedding scheme. It can be expressed as: 
\[ X = [x_{i1}, x_{i2}, \ldots, x_{id}, x_{i0}] \].
The solution corresponds to a vSDN network request including \( d \) virtual nodes and one virtual control node, and \( x_i \) represents a embedding relationship between the virtual node and the physical node. The value is the corresponding number of the underlying physical node after the embedding of virtual node \( n_j \).

For the fitness function, the objective function \( F(X) = f_1 + \omega f_2 \) is calculated in turn to evaluate the pros and cons of the embedding scheme. \( \omega \) is the adjustment factor, the purpose is to keep \( f_1 \) and \( f_2 \) in the same order of magnitude.

The design idea of the proposed two-stage embedding algorithm is as follows: Firstly, the fruit fly population is generated, then the embedding scheme of each individual fruit fly is checked to see if it is feasible. Secondly, the fitness function value is calculated for the feasible scheme and iteratively optimized. Finally, the position of the fruit fly with the smallest fitness value is used as the embedding solution, and perform node embedding and link embedding. In the link embedding process, we assume the physical network does not support path splitting. The flow chart of FOA-vSDNE algorithm is shown in figure 1:

The specific steps are as follow:

**Step 1:** Initialize the experimental network, fruit fly individual \( M \), and iteration number \( NI \). Randomly generate initial position \( X_0 \).

**Step 2:** According to the Eqs. (9-11), it is judged whether each virtual node satisfies the node constraints. According to Eq. (12) and (13), the \( l \) shortest physical paths that satisfy the link constraints are selected by \( K \)-shortest paths.

**Step 3:** Search direction and distance are given to fruit fly individual satisfying the constraints by the following formula.
\[
X_{i+1} = X_i + H \times (2 \times \text{rand}() - 1)
\]
(14)
Where \( \text{rand}() \) produces a random number in [0,1] and the initial search radius \( H \) is 1.

**Step 4:** Adaptive search step size is set, and let's say that \( H = H \cdot \alpha^{NI} \). Where \( \alpha \) is the adaptive selection scaling factor, and the general value range is [0.5, 0.9]. \( NI \) is the current iteration number.

**Step 5:** We calculate the odor concentration value \( C_i = F(S_i) \), that is the fitness value, and finish sorting for \( C_i \).
Step 6: The best and worst odor concentration values and the corresponding position index values are recorded and saved.

\[
\begin{align*}
C_{\text{best, old}} &= C_{\text{best, i}} \\
X_{\text{best}} &= X(C_{\text{best, index}})
\end{align*}
\]

\[
\begin{align*}
C_{\text{worst, old}} &= C_{\text{worst, i}} \\
X_{\text{worst}} &= X(C_{\text{worst, index}})
\end{align*}
\]

(15)

Where \(C_{\text{best, old}}\) and \(C_{\text{worst, old}}\) represent historical best and worst odor concentration values, respectively. \(C_{\text{best, i}}\) and \(C_{\text{worst, i}}\) represent contemporary best and worst odor concentration values, respectively. \(C_{\text{best, index}}\) and \(C_{\text{worst, index}}\) represent the contemporary optimal and worst individual position index, respectively. \(X_{\text{best}}\) and \(X_{\text{worst}}\) represent the contemporary optimal and worst individual position, respectively.

Step 7: The mean and variance of the population odor concentration is calculated.

\[
\overline{C} = \frac{\sum_{i=1}^{M} C_i}{M}
\]

(16)

\[
C_\sigma^2 = \frac{\sum_{i=1}^{M} (C_i - \overline{C})^2}{M}
\]

(17)

Step 8: The odor concentration threshold \(C_0\) is set. According to the relationship between \(C_\sigma^2\) and \(C_0\), update the population position. There are two ways to update as follows:

\[
X'_i = X_i + X_{\text{best}}
\]

(18)

\[
X'_i = X_i - X_{\text{worst}}
\]

(19)

Eq. (18) indicates that if \(C_\sigma^2 > C_0\), the attraction operation is selected to update the group position, and Eq. (19) indicates that if \(C_\sigma^2 < C_0\), the exclusion operation is selected to update.

Step 9: Iteratively perform Step 2-6, and go to Step 10 when the maximum number of iterations is reached. Otherwise, perform Step 7-8.

Step 10: The optimal solution \(X\) is obtained. When the \(X\) satisfies the resource requirement of the vSDN request, the node embedding is performed to \(X\) firstly, then the link embedding is performed by the \(K\)-shortest paths. Output the embedding result, otherwise the embedding fails to end.

3. The Simulation Results

In order to verify the effectiveness of the proposed algorithm, we refer to the improved Salam network topology random generation algorithm in the experiment. The underlying physical SDN network and vSDN requests are generated by the MATLAB. The experimental parameters are set in accordance with the Ref. [4]. The underlying physical SDN network topology is composed of 100 physical nodes and 400 links, randomly and evenly distributed. The node CPU resource, FTSP resource and link bandwidth resource are uniformly distributed in \([0,100]\), and the link delay is uniformly distributed in \([5,20]\). We assume that the arrival is a Poisson process having mean of 5 requests every 100 time units and the life-time of each vSDN request follows an exponential distribution with a mean of 200 time units. Each vSDN request contains a virtual control node and several virtual nodes, and the number of virtual nodes is uniformly distributed in \([3,10]\). The virtual control node directly connects each virtual node to realize the management and control function. The virtual node CPU resource and FTSP resource, virtual link and virtual control link bandwidth resource is uniformly distributed in \([0,30]\). The virtual control node resource requirement is set as 0. The simulation is run for 50000 time units, each time unit is 1 ms. The improved fruit fly algorithm parameters are set as follows: \(M=20\), \(N=30\), \(\alpha = 0.9\), \(H=1\), \(C_\sigma^2=0.0001\), \(\eta = 0.5\).

Each algorithm performs performance evaluation and analysis from four aspects: vSDN request acceptance rate, overall load imbalance, average control delay, and average running time on the same network topology. We contrast our algorithm with RW-BFS algorithm and G-SP algorithm. The Ref. [5] proposed the breadth first search (RW-BFS) algorithm. Firstly, the virtual controller node was embedded. Then the virtual node was embedded centered on the controller node. The candidate physical node set was constructed by the traceback according to the virtual node and the physical node sorting.
G-SP algorithm took a method of controller placement randomly, and embeded nodes by a greedy strategy and embeded links by the shortest paths method.

In Figure 2, we compare the overall load imbalance of the underlying network changes with time under different algorithms. The figure shows that the overall load imbalance of the underlying network by FOA-vSDNE is the smallest, maintaining around 0.187. Compared with the G-SP algorithm (about 0.252) and the RW-BFS algorithm (about 0.235), it is reduced by 34.76 % and 25.67 %, respectively. In Figure 3, we compare the average control delay changes with time under different algorithms. The figure shows that FOA-vSDNE algorithm (30.5~32.9 ms) is occasionally slightly worse in performance. It is smaller than the G-SP algorithm (36.3 ms) overall, slightly larger than the RW-BFS algorithm (29.8~31.6 ms).

Figure 4. vSDN request acceptance rate
In Figure 4, we compare the vSDN request acceptance rate changes with time under different algorithms. The figure shows that due to the rich underlying resources in the initial stage and there are few vSDN requests, thereby the embedding acceptance rate is high. With the increase of time, the request acceptance rate of FOA-vSDNE algorithm is significantly higher than that of RW-BFS algorithm (about 0.678) and G-SP algorithm (about 0.538), which is improved by 25.52 % and 58.18 %, respectively. In Figure 5, we compare the average running time under different algorithms. The figure shows that the G-SP algorithm has the shortest average running time (about 43 ms). The FOA-vSDNE algorithm (about 50 ms) is close to it, which is much smaller than the average running time of the RW-BFS algorithm (about 98 ms).

Figure 5. Average running time

4. Conclusion
In this paper, we considered using improved fruit fly optimization algorithm and K-shortest path method to solve the embedding optimization problem and obtained approximate optimal allocation, which improved network performance to a certain extent.

5. References
[1] Sun C, Bi J, Zheng Z, et al. Nfp: Enabling network function parallelism in nfv[C]//Proc.of the Conference of the ACM Special Interest Group on Data Communication. ACM, 2017: 43-56.
[2] Capelle M, Abdellatif S, Huguet M J, et al. Online virtual links resource allocation in Software-Defined Networks[C]/2015 IFIP Networking Conference (IFIP Networking). IEEE, 2015: 1-9.

[3] Rosa R V, Rothenberg C E, Madeira E. Virtual data center networks embedding through software defined networking[C]/2014 IEEE Network Operations and Management Symposium (NOMS). IEEE, 2014: 1-5.

[4] Li R Z, Wu Q B, et al. On the optimal approach of survivable virtual network embedding in virtualized SDN[J]. IEICE Transactions on Information and Systems, 2018, 101(3): 698-708.

[5] Cheng X, Su S, Zhang Z B, et al. Virtual network embedding through topology-aware node ranking[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(2): 38-47.