SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2016, Vol. 59 092301:1–092301:10
doi: 10.1007/s11432-016-5574-0

A multipath resource updating approach for distributed controllers in software-defined network

Xiaochun WU[1,2], Chunming WU[1]*, Changting LIN[1], Qiang WU[1,3] & Bin WANG[1]

[1] *Computer Science College, Zhejiang University, Hangzhou* 310027*, China;*
[2] *Institute of Network and Communication Engineering, Zhejiang Gongshang University, Hangzhou* 310027*, China;*
[3] *ZTE Central R&D, ZTE Corporation, Nanjing* 210012*, China*

Received October 16, 2015; accepted December 19, 2015; published online June 27, 2016

Abstract Finding effective ways to collect the usage of network resources in all kinds of applications to ensure a distributed control plane has become a key requirement to improve the controller's decision making performance. This paper explores an efficient way in combining dynamic NetView sharing of distributed controllers with the behavior of intra-service resource announcements and processing requirements that occur in distributed controllers, and proposes a rapid multipathing distribution mechanism. Firstly, we establish a resource collecting model and prove that the prisoner's dilemma problem exists in the distributed resource collecting process in the Software-defined Network (SDN). Secondly, we present a bypass path selection algorithm and a diffluence algorithm based on Q-learning to settle the above dilemma. At last, simulation results are given to prove that the proposed approach is competent to improve the resource collecting efficiency by the mechanism of self-adaptive path transmission ratio of our approach, which can ensure high utilization of the total network we set up.

Keywords multipath, Pareto optimal, Q-learning

Citation Wu X C, Wu C M, Lin C T, et al. A multipath resource updating approach for distributed controllers in software-defined network. Sci China Inf Sci, 2016, 59(9): 092301, doi: 10.1007/s11432-016-5574-0

1 Introduction and motivation

Software-defined networking (SDN) [1, 2] allows network administrators to manage network services through abstraction of lower level functionalitie. Presently, the reference implementation for SDN architecture is creating a performance bottleneck and is a single point of failure in large networks [3]. High availability and maintaining low response time are among the reasons that a network needs multiple controllers [4]. Therefore, many researchers are trying to provide a scalable yet efficient solution to distributed SDN network architecture for management, such as [5, 6]. Another aspect, Ref. [7] describes its experience in the design of HybNET, which is a framework for automated network management of a hybrid net infrastructure. Ref. [8] describes a FlowN architecture that gives each tenant the illusion of its own address space, topology, and the controller, and leverages database technology to efficiently store and manipulate mappings between virtual networks and physical switches. However, none of these methods have settled the issue completely. As we know, many management tasks such as traffic accounting, traffic engineering, load balancing and performance diagnosis [9, 10] all rely on accurate and

* Corresponding author (email: wuchunming@zju.edu.cn)

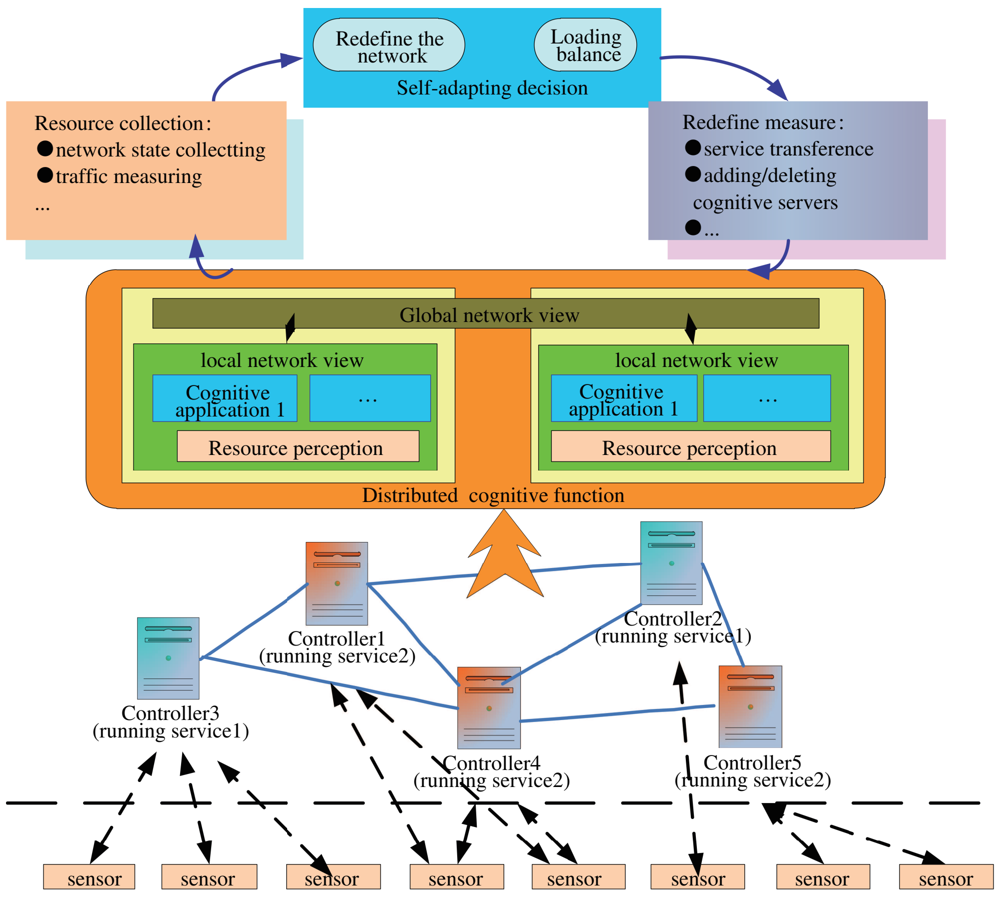


Figure 1 (Color online) The cognitive ability of distributed controllers in SDN.

timely apperceiving of a large variety of network resources at different time-scales. However, there is little research about the resource collecting and optimal management mechanism of multiple controllers on SDN [11]. Only Ref. [12] identifies problems with the current state-of-the-art network configuration and management mechanisms and demonstrates how SDN can improve common network management tasks. And most of the previous work is about network measurements [13–15] that have explored several measurement primitives at switches/routers for measurement tasks [10, 16]. We consider that a SDN distributed controller needs to manage more resources carefully in order to ensure accuracy behavior of traffics, so does the result of resource politic collecting. The controllers managing efficiency [17] keep close to the frequency and content of collection, therefore, a solution to increases the processing capacity of the controller without obvious additional resource management communication expenses is needed. We get some enlightenment from the experience in the design of [18], which provides a scalable yet efficient solution to distributed SDN network management, that manages and coordinates among distributed SDN controllers. But the NetView it puts forward is limited for specific applications such as for the purpose of improving controllers' load. We hope to find a more scalable NetView and some optimal methods to dynamically update its view resources, which can be great of help to manage the SDN networking. In order to obtain timely and accurate information during the course of collection, information collection programs are deployed on each switches/router in the SDN network. And the controller shouldn't connect with any forwarding node to obtain the current resource information. Thus the distributed resource collector should be added to collect resource information between the controller and all kinds of switches/routers. Nowadays, a large SDN network is partitioned in multiple information collecting domains [19]; each distributed controller is responsible for local information collection and part of resource for other controllers in order to enforce global policies.

More flexible SDN applications and optimized services are the major and minor premises on which timely and accurate information collection are based. The information we care about includes local and global topology information, service state etc. Therefore, deployment of soft sensors in physical forwarding nodes is taken into account in the proposed scheme. Besides, cognitive [20, 21] server is added to complete collection and processing network resource in distributed controllers. Owing to the high impact on deploying the location of cognitive server, we should consider the cost of connection between the soft sensor and cognitive server as well as the cost of processing. With the expansion scale of the forwarding plane, a distributed architecture of controllers with cognitive function has been deployed as shown in Figure 1. The local network view needs to be consistent [22], which contains the information of the topology, resource statistics, traffic cost and so on. And a reliable transmission path from one distributed controller to another needs careful consideration. Besides, the transmission overhead and the continuity are also challenging. On one hand, the convergence time of global network view can be released. On the other hand, the soft sensors and the distributed controllers will both benefit from reliable transmission path. The resource information needs to be sent out in time and resources should avoid being repeatedly transmitted, which reduces the cost of the whole network. It also plays a crucial role in reducing the computation cost of regional cognitive function.
The distributed controller needs to timely report the local view information it collected to the other distributed controller. Even in a service, as shown in Figure 1, the functions of each point of distributed controllers are not symmetrical, a traditional pub-sub system or some sort of broadcast protocols are not suitable for such architecture of distributed controllers. Therefore rapid and uninterrupted transmittal paths between these distributed controllers are very important, which can bring benefit to information accepter and the sender itself. There are two ways to achieve this. One is forwarding the packets from switches in a timely manner to the controller for reporting resource information, and the other way is cutting down the transmittal overhead and computational overhead from reducing duplication of transfer. Ref. [23] also shows a load balancing scheme with hop-by-hop routing, by using the burstiness features of flows to make sure that the packets of the same flow arrive at the receiving end in order. Therefore, multiple paths for updating the view information are suitable in our design. The view sharing models will enable us to dynamically build information transfer paths according to the network environment and the utilization of network resources at the moment. More specifically, the contributions of this paper are summarized as follows:

(1) This paper proposes, to the best of our knowledge, a cognitive framework of muti-distributed view gathering scheme in SDN.

(2) This paper presents a rapid multi-path distribution mechanism that enables timely NetView information to be consistent among muti-distributed controllers.

(3) Our work differs from previous related works in that we offer a more scalable NetView updating methods that takes the dynamic characteristics of controllers and load balance of the control plane into consideration instead of just increasing the cognitive information processing power of controllers.

This paper is structured as follows. The problem of prisoner's dilemma in cooperative collecting information dispatching is presented in Section 2. Section 3 discusses the rapid multi-path distribution mechanism in detail. Section 4 presents the rapid multi-path distribution mechanism which consists of a bypass path selection algorithm and a diffluence algorithm based on Q-learning. Section 5 gives out the experimental results and analysis. The conclusion and potential future work are listed in Section 6.

2 The problem of prisoner's dilemma in cooperative collecting information dispatching

As mentioned above, consistency of network view of distributed controllers in SDN lies on the timely and effectively collected information, which consumes a lot of bandwidth, computation and storage resources. So each controller is always interested in things such as the conservation of resources, the improvement of the efficiency and is apt to decline to help other distributed controllers in transferring collected informa-

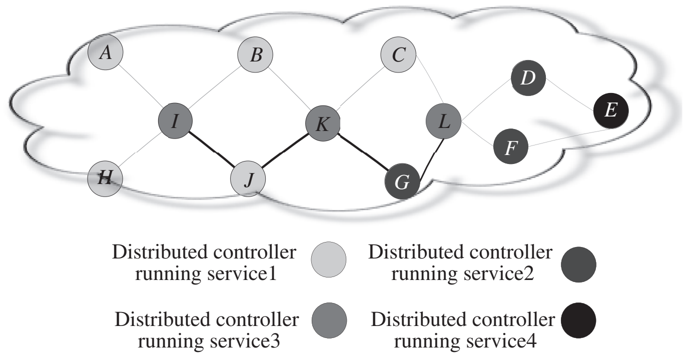

A
B
C
D
E
I
K
L
F
H
J
G
Distributed controller running service1
Distributed controller running service2
Distributed controller running service3
Distributed controller running service4

Figure 2 The location of distributed controller.

tion. Thus each distributed controller running different service is apt to be rebellious as follows, which is termed as not cooperative behavior.

Definition 1. The set of collection nodes is $C : \{1, 2, \ldots, i, \ldots, j, \ldots, n\}$, the information collection of node i is N_i. The discrete time slot $(t = 1, 2, \ldots, n)$. The set of agent node is U, $U \cup C \neq i$. The number processing node i in U at time slot is $s(t)$, the number of i sending information to U at time slot is $k(t)$. b is the income of I, which helps the set of agent U. c is the overhead of I, which helps the set of agent U. b and c are constants. Therefore, the revenue function of node i according to cooperative processing is given by

$$U_i(t) = bs_i(t) - ck_i(t). \tag{1}$$

As any node in a set of U hopes to obtain the maximal profit according to the cooperative behaviour, the node i will choose optimal treatment strategy. We define $k_i'(t)$ as its maximal revenue function. The policy of maximal $k_i'(t)$ is as follows:

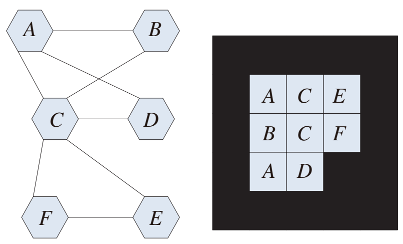
$$k_i'(t) = \arg\max_{f_i \geqslant 0} U_i(s_i'(t), k_i(t)), \tag{2}$$

whatever the value of b, c and $s_i(t)$ when $k_i(t) = 0$, $U_i(t)$ can get the maximal value. Then the node i in the network $(i \in C)$ will choose the policy of $k_i'(t) = 0$, so for any i, $s_i(t) = 0$. Therefore we can get $U_i(t) = 0$. The above results mean that any forwarding node in this network will never help other forwarding nodes to send or process the information they collected. The throughput of collecting and processing will be zero, it is a problem of prisoner's dilemma.

As mentioned before, there are many researches about game theory to settle down this problem, which is against the whole system's self-profit behaviour. In this paper, we put forward incentive strategy based on cooperation by which each controller benefits not only itself but also others, which is a double winning strategy.

3 Multipath rapid distribution mechanism

The rapid distribution mechanism of paths consists of the selection algorithm of the bypass paths and the distribution algorithm of the information. As we know, the collected information should be sent from one distributed controller to the other. We suppose that there are only two pieces of paths that can be chosen by the collection node, one is the shortest route and the other is the bypass route. For example, as shown in Figure 2. A can choose the shortest route IJK or the bypass route IBK. As usual, we are prone to choose the shortest route to send our information, but if link JK is busier than the bypass route, we should choose IBK to avoid packet loss. Packet reordering caused by multi-path transmitting will be settled down according to the tag and cache in the accepted node. We denote the probability of the shortest route we choose by α_i^t, then the probability of bypass route is $1 - \alpha_i^t$. The probability of information arriving in node j is λ_i^t. The probability of information leaving from source



$$\begin{bmatrix} A & C & E \\ B & C & F \\ A & D & / \end{bmatrix} \leftarrow \begin{bmatrix} A & \underline{C} & E \\ B & \underline{C} & F \\ A & D & / \end{bmatrix} \leftarrow \begin{bmatrix} \underline{\underline{A}} & * & \underline{E} \\ \underline{\underline{B}} & * & \underline{F} \\ \underline{\underline{A}} & \underline{\underline{D}} & / \end{bmatrix}$$

Figure 3 Network topology description.

node to destination j via the shortest path in t is

$$\beta_{lj}^{t} = 1 - \prod_{k_l \in \{\text{shortestpath}\}} (1 - \beta_{k_l}^{t_{k_l}}), \quad l \in R_j^t,$$

where R_j^t is the node set from source to destination j in time slot t. Then, leaving from the source node to destination j via the bypass path in t is

$$\omega_{lj}^{t} = 1 - \prod_{g_l \in \{\text{passbypath}\}} (1 - \beta_{g_l}^{t_{g_l}}), \quad l \in R_j^t,$$

the state information of all nodes that arrive at their destination via the shortest route in t is $\sum_{l \in R_l^t} \alpha_l^t \lambda_j^t \times (1 - \beta_{lj}^t)$, so the bypass route will be $\sum_{l \in R_j} (1 - \alpha_l^t)\lambda_j^t(1 - \omega_{lj}^t)$, the information sent out from the set of node R_j^t in t, the throughput of j is $\theta_j = [\sum_{l \in R_j} \alpha_l^t \lambda_j^t (1 - \beta_{lj}^t) + \sum_{l \in R_j} (1 - \alpha_l^t)\lambda_j^t(1 - \omega_{lj}^t)] * H_j^t$, where H_j^t is the state information that should be transferred in j at time of t. Therefore, the throughput of state information collected in the whole network is $\theta = \int_{t=0}^{\infty} \sum_{j=1}^{N} \theta_j$. Our aim is to find the best split point for all forwarding nodes.

$$(\alpha_1, \alpha_2, \ldots, \alpha_N) = \arg\max_{\alpha \in [0,1]} \int_{t=0}^{\infty} \sum_{j=1}^{N} \left[\sum_{l \in R_j} \alpha_l^t \lambda_j^t (1 - \beta_{lj}^t) + \sum_{l \in R_j} (1 - \alpha_l^t) \lambda_j^t (1 - \omega_{lj}^t) \right] * H_j^t \, \mathrm{d}t.$$

Definition 2. $\theta(\alpha_i, \alpha_{-i})$ is the collected information transmission throughput for the whole network. $(\alpha_1^*, \alpha_2^*, \ldots, \alpha_{-i}^*, \ldots, \alpha_N^*)$ is a Nash equilibrium point. If and only if $\forall i \in N$, $0 \leqslant \alpha_i \leqslant 1$, we have $\theta(\alpha_i^*, \alpha_{-i}^*) > \theta(\alpha_i, \alpha_{-i}^*)$.

It is a game system composed of N collectors, and the Nash equilibrium is a steady state. We can't obtain more benefits if the probability of the path any collector chosen deviates from the Nash equilibrium point. We have proved that the function is a strictly concave function. The biggest diversion probability sequence $(\alpha_1^*, \alpha_2^*, \ldots, \alpha_i^*, \ldots, \alpha_N^*)$ for each utility function is the Nash equilibrium point.

The way mentioned above is a compromise of fairness and selfishness between distributed collectors and intra-domain collectors. The cost and interest of each node has been considered in path selection. In this paper, in order to gain the optimum decision, the methods of reinforcement learning have been introduced to improve the routing strategy.

3.1 A bypass path selection algorithm based on Q-learning

According to the description of the topology matrix in network view, the topology matrix should be fixed when there are fixed number of controllers. As Figure 3 shows, node c is the sender, while nodes A, B, D, F and E are receivers.

An intelligent agent starts from node C and its destination is another distributed collector for the same service. The black area is the obstacle which the agent can't pass over. A reward for 100 when the agent arrives at the destination, then return to node C for searching new path to the next destination. In the process of searching, it will get comeuppance of -10 when the agent comes across barriers. Another value of award should be measured by the load of a node.

Q-learning algorithm (Algorithm 1) is shown as follows:

$Q(s,\alpha)$ is the function of Q-learning, storing it according to its exchange from matrix to table, the size of it is $S \times A$, i.e., the result of Cartesian product. The state $s \in S$ denotes the action $\alpha \in A : \{\text{up}, \text{down}, \text{left}, \text{right}\}$. We define the heuristic function $H(s,\alpha) = \sqrt[2]{\sum_{i=1}^{N}(w_{ih}(s'_h))^2}$ which is the Weighted Euclidean distance from the next action to the destination action, it records the environment information when the action α is in the state of s. $h_\alpha(s,H) = \arg\max H(s,\alpha)$, where $h_\alpha(s,H)$ denotes the worst action.

Algorithm 1 A bypass path seletion algorithm based on Q-learning

```
step1 initialize $Q(s,\alpha)$, Model$(s,\alpha)$;$\forall s \in S, \alpha \in A$
step2 do while
step3 record the state of $s$ in time $t$;
step4 the agent choose an action according to greedy algorithm:$\alpha \leftarrow \varepsilon\_greedy(s,Q)$;
step5 perform an action and observe the state of $s'$ and $r$
step6 $Q(s,\alpha) = Q(s,\alpha) + \alpha\{r_t + \gamma \max'_{\alpha \in A}[Q(s',\alpha') - Q(s,\alpha)]\}$;
step7 Model$(s,\alpha) \leftarrow s', r$
step8 $\alpha \leftarrow h_\alpha(s,H)$
step9 if $s', a! \in$ Model random choose $s$, $\alpha$;
step10 otherwise $s', r \leftarrow$ Model$(s,\alpha)$
step11 renew $Q(s,\alpha), H(s,\alpha) : Q(s,\alpha) = Q(s,\alpha) + \alpha\{r_t + \gamma \max'_{\alpha \in A}[Q(s',\alpha') - Q(s,\alpha)]\}; s' \leftarrow S$
step12 judging finished or not, if not then goto step 9;
```

3.2 A diffluence algorithm based on Q-learning

Definition 3. A stochastic game model with n nodes as multi-component system $\langle n, S, A^1, \ldots, A^n, R^1, \ldots, R^n, P\rangle$, which is on behalf of nodes in states pace S, the action space of node i is A^i, joint action space is $A^1 \times \cdots \times A^n$, the reward function of node i is r^i, P is the state transition probability and P is the state transition probability and $P : S \times A^1 \times \cdots \times A^n \to \Delta(s)$ where $\Delta(s)$ is the probability of the state space distribution set.

Definition 4. Multiplexing Policy$s$ is he state set S. s_i becomes two parts $\{s_1, s_2\}$, s_1 denotes the distribution probability, $\alpha_i \in [0:1]$, s_2 represents the information throughput transferred by the node i. An agent should choose one action in $A^i = \{\alpha_i - 2\varepsilon, \alpha_i - \varepsilon, \alpha_i + \varepsilon, \alpha_i + 2\varepsilon\}$. ε is a change factor of the distribution probability. Policy $\pi : s \to A$ is the mapping from states to actions. π_* is the optimal strategy.

Algorithm 2 is shown as follows. There are two kinds of policies in this algorithm, i.e., the current strategy $\pi(s,\alpha)$ and the average strategy $\bar{\pi}(s,\alpha')$, respectively. The optimal action is $\alpha = \arg\max_\alpha Q(s,\alpha')$. We should increase the step δ when it's easy to obtain the maximal Q function, otherwise we should slow down the speed of action such as decrease $\frac{\delta}{\gamma}$. The algorithm revises the strategy $\pi(s,\alpha')$ to average strategy $\bar{\pi}(s,\alpha')$. For instance, when $\sum_\alpha \pi(s,\alpha')Q(s,\alpha) > \sum_\alpha \bar{\pi}(s,\alpha')Q(s,\alpha)$, learning at a speed of δ_1; Otherwise, learning at a speed of δ_h.

4 Experiments and conclusion

A set of numerical experiments is carried out to assess the performance of the bypass path selection algorithm and the distribution algorithm based on Q-learning. We implement these two algorithms by

Algorithm 2 A diffluence algorith based on Q-learning

```
step1: Initialize
    $Q(s,\alpha) \leftarrow 0;\ \ \pi(s,\alpha) \leftarrow \frac{1}{\gamma};\ \ C(s) \leftarrow 0$
step 2: Node $i$ random choose $A^i$ in
    $\{\alpha_i - 2\varepsilon, \alpha_i - \varepsilon, \alpha_i + \varepsilon, \alpha_i + 2\varepsilon\}$.
Renew $Q$ according to $r'(A'|S')$ :
    $Q(s,\alpha) \leftarrow (1-\alpha)Q(s,\alpha) + \alpha[r + \gamma \max_{\alpha' \in A} Q(s',\alpha')]$;
Renew the average policy $\bar{\pi}, j \leftarrow j+1$
    $\forall \alpha \in \bar{\pi}(s,\alpha') \leftarrow \bar{\pi}(s,\alpha') + \frac{1}{j}(\pi(s,\alpha') - \bar{\pi}(s,\alpha'))$
Renew
    $(s,\alpha') \leftarrow \pi(s,\alpha') + \begin{cases} \delta, & \text{if}\, \alpha = \arg\max_\alpha Q(s,\alpha'), \\ -\frac{\delta}{\gamma}, & \text{other}, \end{cases}$    (3)
    $\delta = \begin{cases} \delta, & \delta_1 \sum_\alpha \pi(s,\alpha')Q(s,\alpha) > \sum_\alpha \forall \alpha \in \bar{\pi}(s,\alpha')Q(s,\alpha), \\ \delta_h, & \text{other}. \end{cases}$    (4)
    $\delta_h$ is the rapid increase of the step value, $\delta_1$ is the slow step value $\delta_h = \gamma\delta_1$
step 3: the repeat do step 2 when $\forall i \in N, |\Delta_{s_i}| \leqslant \delta$ algorithm convergence.
```

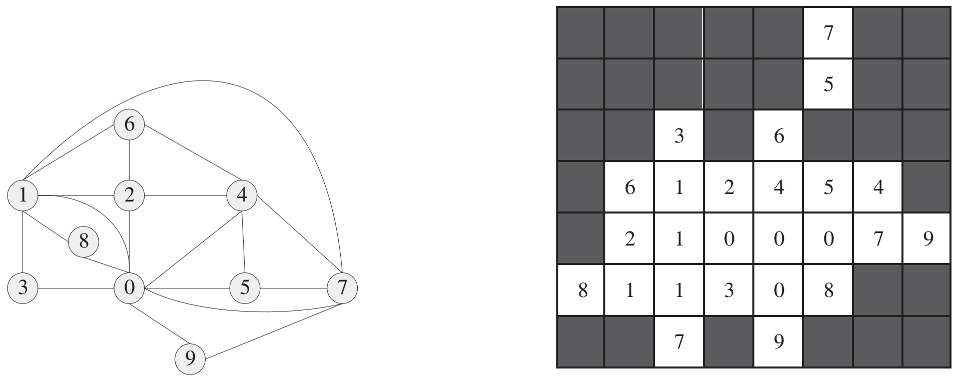


Figure 4 Forwarding network topology and the topology matrix.

C language and employ Matlab together with NS2 to illustrate the results. There are two parts in our experiments. One is the view convergence performance algorithm from the two groups of distributed controllers as senders. The other is the performance variation of information transmission throughput in the network with different scales of collectors. We define ten controllers' nodes for the structure of a network topology as shown in Figure 4. The network topology matrix is as Figure 4 shows [24].

4.1 View convergence performance from two groups of distributed controllers

Firstly, each controller runs the election agreement to determine the location of distributed controllers for one service. The result is that node 1 and 3 belong to one group of distributed controllers, the same as 4 and 5. Then, node 1 will send the resource information to node 3. Node 4 will send its resource to node 5 as Figure 4 shows.

We define the normalization weight of all the nodes in this topology of network as

$$W_{ij} = \begin{bmatrix} 0 & \cdots & 0.8 \\ \vdots & \ddots & \vdots \\ 0.8 & \cdots & 0 \end{bmatrix}, \quad i,j < 10.$$

For us, packet loss rate is the prime important consideration. As Figure 5 shows, the shortest path of 3-1

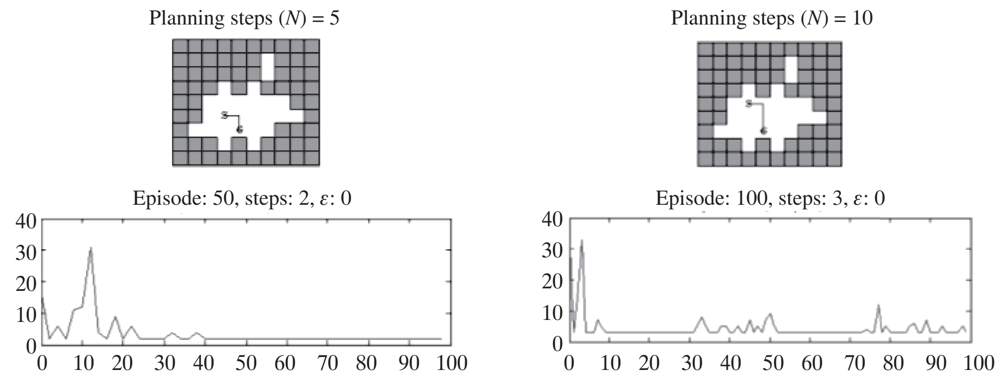


Figure 5 A bypass path selection algorithm based on Q-learning.

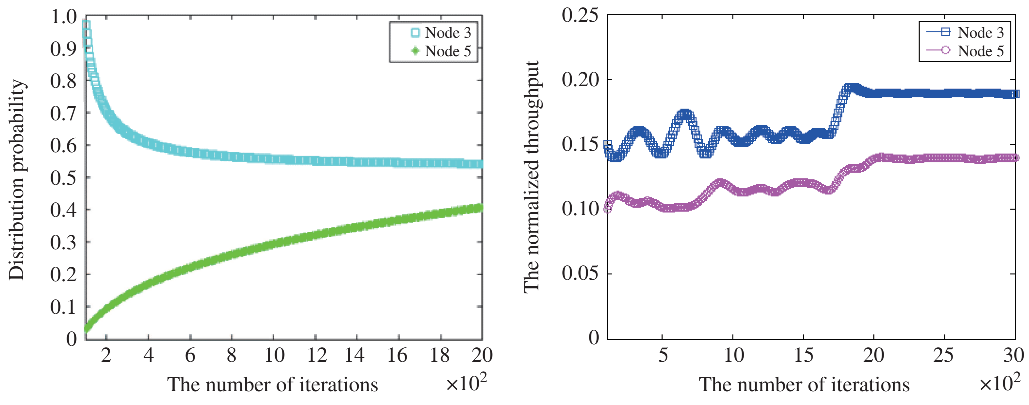


Figure 6 (Color online) The influence of diffluence algorithm on distribution probability and normalized throughput.

and the optimized path of 3-0-1 can be calculated by our Q-learning algorithm when the network load is low. However, another optimized path of 3-0-2-1 can be calculated by our Q-learning algorithm when the network is on condition of heavy load such as the link of 3-1 and 0-1 are in congestion state. It is the same with the optimized path from node 5 to node 4 is 5-0-4, and the shortest path is 5-4. We calculate the weight of 3-1 and 3-0-1 are (0.3,0.6), and the weight of 5-4 and 5-0-4 are (0.4,0.7). The best flow equilibrium point of the two paths to node 3 is (45,55), to node 5 is (38,62).

As shown in Figure 6, the initial distribution probability of node 3 is (0.99,0.01), and the initial distribution probability of node 5 is (0.01,0.99). The distribution probability and throughput change dynamically with the number of iterations. To the final convergence, node 3 and node 5 do a lot of explore work, and then they find the optimal diffluent equilibrium (55,38) according to 1800 iterations.

In the beginning, 1% perception information of the total data sent from node 3 to intermediate node 0. And 99% perception information of the total data is sent from node 5 to intermediate node 0. After a period of reinforcement learning, 55% perception information of the total data sent from node 3 to the competitive node 0 by game, and perception information from node 5 to the competitive node 0 are decreasing to 62%. The experiment results show that node 5 subsumes its own interests, and the perception of information throughput of the whole network is improved, which is brought about the Nash equilibrium.

4.2 With scale of collectors, performance changes by acquisition information throughput in the whole network

We test the bypass path selection algorithm and the diffluence algorithm based on Q-learning algorithm with different scales of collectors using link congestion from (1,3), (2,6), (0,9) and (4,5). Figure 6 shows the influence of throughput in the whole network.

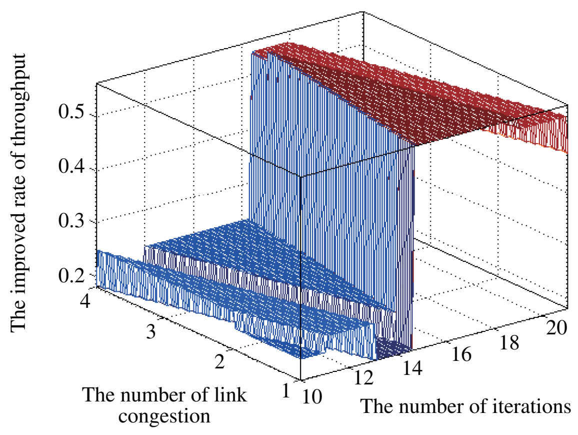


Figure 7 (Color online) The influence of throughput in whole network.

With the reinforced learning algorithm, the convergence of network throughput is increased by 50% and 20%, respectively, regardless of the introduction of diffluence mechanism or not.

As shown in Figure 7, the throughput in the whole network is raised with the increasing cooperation nodes caused by link congestion. But the trend is weakened. Experiments show that the bypass path selection algorithm and diffluence algorithm are competent to improve the network throughput by collecting the resource information in time, which greatly reduces packet losses and the delay in retransmissions.

Acknowledgements This work was supported by National Science and Technology Support Program (Grant No. 2014BAH24F01), National Basic Research Program of China (973) (Grant No. 2012CB315903), Program for Zhejiang Leading Team of Science and Technology Innovation (Grant Nos. 2011R50010-05, 2013TD20).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 ONF White Paper. Software-defined networking: the new norm for networks. Open Networking Foundation, 2012
2 Koponen T, Casado M, Gude N, et al. Onix: a distributed control platform for large-scale production networks. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. USENIX Association Berkeley, 2010. 351–364
3 Yaganeh S H, Tootoonchian A, Ganjali Y. On the scalability of software-defined networking. IEEE Commun Mag, 2013, 51: 136–141
4 Tootoonchian A, Gorbunov S, Ganjali Y, et al. On controller performance in software-defined networks. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association Berkeley, 2012. 10
5 Zuo Q Y, Chen M, Ding K, et al. On generality of the data plane and scalability of the control plane in software-defined networking. China Commun, 2014, 11: 55–64
6 Hu J, Lin C, Li X Y, et al. Scalability of control planes for Software defined networks: modeling and evaluation. In: Proceedings of IEEE 22nd International Symposium on Quality of Service (IWQoS), Hong Kong, 2014. 147–152
7 Lu H, Arora N, Zhang H, et al. HybNET: network manager for a hybrid network infrastructure. In: Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference. New York: ACM, 2013. 6
8 Drutskoy D, Keller E, Rexford J. Scalable network virtualization in software-defined networks. IEEE Internet Comput, 2013, 11: 20–27
9 Kreutz D, Ramos F, Verissimo P. Towards secure and dependable software-defined networks. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2013. 55–60
10 Curtis A R, Mogul J C, Tourrilhes J, et al. DevoFlow: scaling flow management for high-performance networks. ACM SIGCOMM Comput Communn Rev, 2011, 41: 254–265
11 Benson T, Anand A, Akella A, et al. MicroTE: fine grained traffic engineering for data centers. In: Proceedings of the 7th COnference on Emerging Networking Experiments and Technologies. New York: ACM, 2011. 8
12 Kim H, Feamster N. Improving network management with software defined networking. IEEE Commun Mag, 2013, 51: 114–119
13 Yu C, Lumezanu C, Singh V, et al. FlowSense: monitoring network utilization with zero measurement cost. In: Proceedings of the 14th International Conference on Passive and Active Measurement. Berlin/Heidelberg: Springer-Verlag,

2013. 31–41

14 Jose L, Yu M, Rexford J. Online measurement of large traffic aggregates on commodity switches. In: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association Berkeley, 2011. 13

15 Yu M, Lavanya J, Miao R. Software defined traffic measurement with OpenSketch. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation. USENIX Association Berkeley, 2013. 29–42

16 Yuan L, Chuah C N, Mohapatra P. Towards programmable network measurement. Trans Network, 2011, 19: 115–128

17 Heller B, Sherwood R, McKeown N. The controller placement problem. In: Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks. New York: ACM, 2012. 7–12

18 Marconett D, Yoo S J B. FlowBroker: a software-defined network controller architecture for multi-domain brokering and reputation. J Netw Syst Manag, 2015, 23: 328–359

19 Hassas Yeganeh S, Ganjali Y. Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks. New York: ACM, 2012. 19–24

20 Thomas R W, Friend D H, DaSilva L A, et al. Cognitive networks. In: Arslan H, ed. Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems. Netherlands: Springer, 2007. 17–41

21 Femminella M, Francescangeli R, Reali G, et al. An enabling platform for autonomic management of the future Internet. IEEE Network, 2011, 25: 24–32

22 Reitblatt M, Foster N, Rexford J, et al. Consistent updates for software-defined networks: change you can believe in! In: Proceedings of the 10th ACM Workshop on Hot Topics in Networks. New York: ACM, 2011. 7

23 Chen F, Wu C M, Wang B, et al. Dynamic load distributed with hop-by-hop forwarding based on max-min one-way delay. Sci China Inf Sci, 2014, 5: 062310

24 Wu X C, Wu C M, Wang B, et al. Network view and cognitive mechanism for virtual network resource management based intelligent. Chin J Electron, 2014, 23: 574–578