Qiu *et al. EURASIP Journal on Wireless Communications and Networking* (2016) 2016:282
DOI 10.1186/s13638-016-0779-x

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Sleeping mode of multi-controller in green software-defined networking

Chao Qiu[1*], Chenglin Zhao[1], Fangmin Xu[1] and Tianpu Yang[2]

Abstract

Although promising the formidable configuration, vigorous evolution, and satisfactory performance, software-defined networking (SDN) is in its initial period all the same. Some essential issues still remain not completely resolved, and the scalability of control plane is the most intractable one with the explosive increase of network traffic. To address this issue, many researchers have proposed multiple controllers to realize logically centralized control layer. Our previous research has proposed multi-controller load balancing approach called HybridFlow. In this paper, taking advantage of HybridFlow, we propose an *M-N* policy multiple-controller sleeping mode by switching off redundant controllers when the system is in the light traffic condition. We use queuing theory to model the operation procedure of controllers and formulate the energy consumption management issue as a *0-1* integer linear programming model. Through turning off the redundant controllers when the system is in the scenario of light traffic, the total energy consumption of the whole system can be cut down. Simulation results reveal that the proposed *M-N* policy multiple-controller sleeping mode achieves superior energy efficiency compared to no sleeping mode and *N* policy sleeping mode. However, it introduces tolerable time delay.

Keywords: Multi-controller, Sleeping mode, Energy efficiency, Software-defined networking

1 Introduction

Radar networks have achieved great success with more fragile waveform design and diversity. Waveform diversity is the technology that allows one or more sensors on board a platform to automatically change operating parameters, e.g., frequency, gain patten, and pulse repetition frequency (PRF) to meet the varying environments [1, 2]. It has long been recognized that judicious use of properly designed waveforms, coupled with advanced receiver strategies, is fundamental to fully utilize the capacity of the electromagnetic spectrum [3]. It is fueling a worldwide interest in the subject of waveform design and the use of waveform diversity techniques [4, 5]. Based on the definition of waveform diversity, radar networks need to gain the characteristics of targets and environmental information interactively and change firing waveform flexibly so as to optimize performance of detection, tracking, and anti-interference. So that it is necessary to improve the flexibility of software and hardware in radar networks. SDN has a potent advantage in the flexibility of networks.

With the TCP/IP protocol, Internet has achieved great success. However, burgeoning trends in the information and communication technologies (ICT) domain, especially along with the appearance of emerging technologies such as cloud computing and big data analysis, these technologies have added further demands on the flexibility and agility of computer networks. To solve current issues and embarrassment of Internet, some clean-slate architectures have been suggested by research communities to build the future Internet architecture. SDN is touted as one of the most promising paradigms [6]. It is an approach to building computer networks that separates and abstracts elements of these systems, which promises to dramatically simplify network control and provide a flexible platform to implement a series of applications [7]. One important traits of SDN is that it allows logical centralization of feedback control with better decisions based on global network view and cross-layer information via a famous protocol called Openflow [8].

*Correspondence: zjkchouchao@163.com
[1]School of Information and Communication Engineering, Beijing University of Post and Telecommunications (BUPT), Xi Tu Cheng Road 10, 100876 Beijing, China
Full list of author information is available at the end of the article

In the early days of Openflow, only a single controller is proposed. With the increase of network scale and scalability requirements [9], the single controller is the choke point of SDN. In Openflow, the important feature is flow table, which makes the control plane flexibly configure the data plane. The switches in the data plane become "stupid" packet forwarding devices. However, when a new flow arrives which is strange for this switch, the switch will send this "novel" packet to the controller by a packet-in message [8]. The controller makes decisions using current global network view to deal with the received packet-in messages, which also stores the forwarding decision in the flow-mod message to reply the switch. Fatally, a single controller cannot satisfy the demand of large-scale network. In the [10], the evidence is given. It analyzes the real traffic of China Education Network and finds that the maximal number of flows per second is about 3 million. In this situation, the controller is in the enormous pressure due to the capacity limitation of flow table in TCAM. The throughput of NOX with a single thread is about 21,000 messages per second. Using the multi-threading technique, Maestro [11] improves the controller's performance, but the throughput of a single controller is still far away from 3 million. Thus, it can be seen that improving the scalability and reliability of control plane is extremely urgent.

However, using multiple controllers to constitute a logically centralized control plane is a famous solution. The issues of multiple controllers have been studied in the literature [12, 13]. In [12], Ethane mainly applies multiple controllers to provide fault tolerance. In [14, 15], researchers have proposed distributed controller implementations called HyperFlow and Onix. These implementations use multiple controllers to manage the entire network and exchange cross-layer information to make sure the consistency of global information, which achieves the logical centralization in the domain and improves the scalability of control plane. The author of [13] proposes BalanceFlow, which is a controller load balancing architecture for wide-area network. BalanceFlow uses CONTROLLER X action for switches and elects a super controller to flexibly adjust the flow request.

Obviously, with the appearance of multiple controllers, the energy consumption of SDN will increase observably. Energy efficiency of operating an Internet-scale system is a growing concern due to the increasing energy costs. It is reported that the Information and Communication Technology (ICT) sector is responsible for up to 10% of global energy consumption, and 51% of that is attributed to telecommunication infrastructure and data centers. The US Environment Protection Agency (EPA) estimates that servers and data centers could consume 100 billion kilowatt hours at a cost of 7.4 billion dollars per year, which is equivalent to the output of 30 nuclear power plants. Hence, the energy consumption of networks cannot be ignored. As an emerging future network architecture, we need to pay attention to the research of energy consumption in SDN foreseeingly. In [10], it proposes a dormant multi-controller model which allows parts of idle controllers to enter the dormant state for saving system cost, and when the queue length in the system increases and surpasses threshold N, the system activates all dormant controllers to provide service that we called as N policy sleeping mode in the following paragraph.

In this paper, based on previously proposed multi-controller load balancing approach called HybridFlow, we propose an M-N policy multiple-controller sleeping mode by switching off redundant controllers when system is in the light traffic condition. We use queuing theory to model the operation procedure of controllers and formulate the energy consumption management problem as an *0-1* integer linear programming model. We propose the M-N policy, which works as follows: when the queue length in HybridFlow system is under light condition and super controller detects the whole queue length of multiple controllers hitting the threshold M, then starts algorithm to close some appropriate controllers based on the principle of minimum cost function. Selected controllers transfer their loads to other controllers according to the principle of load balancing and minimum transmission overhead and time delay and go to sleeping mode. Then, the sleeping controllers return to work when N packets have been accumulated during the sleeping period. Compared with N policy sleeping mode, our proposed mode can improve energy efficiency. However, with improving energy efficiency, M-N policy sleeping mode introduces more time delay. In addition, based on simulation, with different M and N, there is the fundamental tradeoff between energy consumption and time delay.

The rest of this paper is organized as follows. In Section 2, the system model is given. On this foundation, the energy efficiency issue to resolve is formulated. The sleeping algorithm is introduced in detail in Section 3. Simulation results are presented and discussed in Section 4. Finally, we conclude this study in Section 5.

2 System model

In this section, we briefly present an overview of HybridFlow, the network topology, energy consumption model, and queuing theory model.

2.1 Overview of HybridFlow

As shown in Fig. 1, HybridFlow's control plane consists of one super controller and several clusters, where each cluster contains some controllers called cluster controllers. Note that we only indicate a finite number of controllers in Fig. 1, and switches in the data plane are ignored.

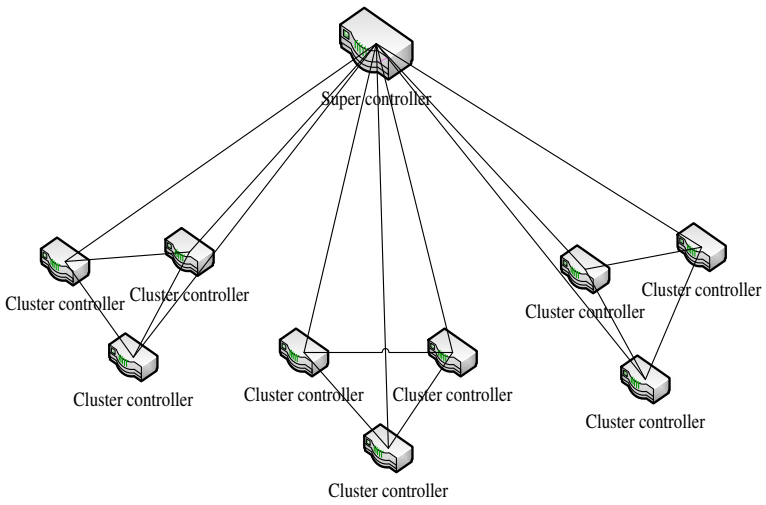

Fig. 1 HybridFlow architecture

The controller enables to communicate with other controllers which are in the same cluster directly, and controllers which belong to different clusters connect with others via super controller. Under such a communicated mechanism, network status is synchronized between all the cluster controllers, such as global view information, topological information, and routing paths. Super controller is a special device which has the following function modules: forwarding, storage, calculation, timing, etc. Its main functions are to collect the load status from cluster controllers, correlatively calculate and forward. In addition, super controller connects the cluster controllers from different clusters respectively and monitors the load status of each cluster controller.

2.2 Network topology

HybridFlow consists of a super controller and x^*y cluster controllers, and we set $X=\{1,...,x\}$ which denotes the set of x clusters, $Y=\{1,...y\}$ which determines the set of y cluster controllers in one cluster, and cluster controller j in i cluster is denoted as cluster controller$_{ij}$, for short as CC_{ij}. Without loss of generality, we assume the number of cluster controllers in each cluster is the same. The approach could be extended to other scenarios. The cluster can be modeled as the directed full connection graph $G=\{V, E\}$, where we let the elements in set V denote the nodes in the graph G and the elements in set E represent the edges in the graph G. In addition, we assume that all of cluster controllers enable to go to sleep when receiving the centralized configuration command message from super controller, and the sleeping cluster controller enables to awaken itself when its queuing length exceeds a certain value.

2.3 Energy consumption model

Warkozek et al. [16] introduces the energy model for servers, which considers the server as a black box modeling, depending on a few parameters and inputs, whose advantage is the possibility to be generalized with an acceptable error of estimation. And in [16], the energy consumption of servers is expressed as a nonlinear function of the CPU usage of the server, presenting in (1). In this case, server's power is formulated as a quadratic function of CPU usage.

$$\mathbf{y} = P_{\text{idle}} + (P_{\max} - P_{\text{idle}}) * (2u - u^r) \tag{1}$$

where P_{idle} is the energy consumption when there is no load; $P_{\max}$ is the energy consumption when there is full load; u is the percentage of CPU usage due to the load; and r is a parameter whose value is found by experimental measurements, and the proper value of r should minimize the quadratic error of estimation. But in this paper, we just need to know the relative value of energy consumption, which is sufficient to determine the effect of the proposed algorithm in this paper, and there is no need to calculate the absolute value of energy consumption accurately. Based on the above consideration, we assume $r = 2$ in this paper.

2.4 Queuing theory model

We use queuing theory to model the arrival, processing, and departure of packet-in message in cluster controllers with shutting down time and starting time as shown in Fig. 2.

In addition, we consider each of controllers receives packet-in messages from the data plane independently, thus they can be modeled as an independent single service

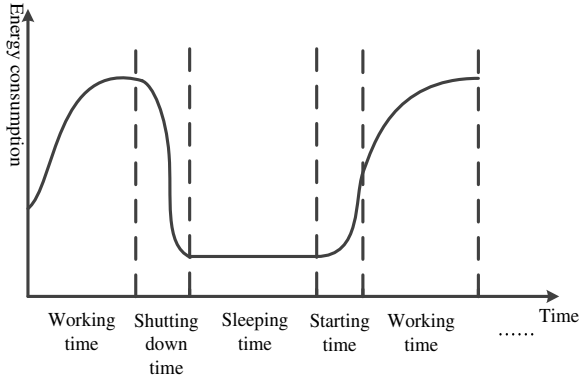

Fig. 2 Energy consumption of the cluster controller with shutting down time and starting time

window mixed system queuing model $M/M/1/m$, where the time interval of messages arriving at CC_{ij} follows the negative exponential distribution with parameter λ_{ij}, the time of CC_{ij} processing one message follows the negative exponential distribution with parameter μ_{ij}, and the maximum number of queuing packets in each cluster controller is m. If there are m messages queuing in the cluster controller, the newcome message will be discarded immediately, which is called instant refused system. If service rate is less than arrival rate ($\mu_{ij} < \lambda_{ij}$), the departure rate is less than the arrival rate which will result in the congestion and packet loss in the control plane. In this paper, we consider a multiple-controller sleeping mode by switching off redundant controllers when system is in the light traffic condition. So we set $\mu_{ij} > \lambda_{ij}$ in the paper. State-flow Markov chain is shown in Fig. 3. Under the control of super controller, there are x^*y queuing models in HybridFlow.

In the state 0: $\lambda_{ij}{}^*p_0{=}\mu_{ij}{}^*p_1$, so $p_1{=}\ (\lambda_{ij}/\mu_{ij})^*p_0{=}\rho_{ij}{}^*p_0$, where $\rho_{ij}{=}\lambda_{ij}/\mu_{ij} < 1$;

In the state 1: $\lambda_{ij}{*}p_1 = \mu_{ij}{*}p_2$, so $p_2 = (\lambda_{ij}/\mu_{ij}){*}p_1 = \rho_{ij}$ $*p_1 = \rho_{ij}^2 * p_0$;

...

In the state $m-1$: $\lambda_{ij}{}^*p_{m-1}{=}\mu_{ij}{}^*p_m$, so $p_m{=}$ $(\lambda_{ij}/\mu_{ij})^*p_{m-1}{=}\rho_{ij}^{m*}p_0$; where p_k, $k = 0, 1, \ldots m$ is the probability of k messages in system when it is in steady state. We have (2)

$$\sum_{k=0}^{m} p_i = \sum_{k=0}^{m} \rho_{ij}^k p_0 = \frac{1-\rho_{ij}^{m+1}}{1-\rho_{ij}} p_0 = 1, \tag{2}$$

and we can deduce (3) and (4):

$$\mathbf{p}_0 = \frac{1-\rho_{ij}}{1-\rho_{ij}^{m+1}} \tag{3}$$

$$\mathbf{p}_k = \frac{1-\rho_{ij}}{1-\rho_{ij}^{m+1}} \rho_{ij}^k \qquad k = 1, 2, \ldots, m \tag{4}$$

and the expectation of messages waiting delay in the CC_{ij} is derived in (5).

$$\begin{aligned}
\mathbf{L}_{q_{ij}} &= \sum_{k=0}^{m-1} k p_{k+1} \\
&= \rho_{ij}^2 p_0 \sum_{k=1}^{m-1} k \rho_{ij}^{k-1} \\
&= \rho_{ij}^2 p_0 \left(\frac{1-\rho_{ij}^m}{1-\rho_{ij}} \right)' \\
&= \rho_{ij}^2 p_0 \frac{1-\rho_{ij}^{m-1}\left[m-(m-1)\rho_{ij}\right]}{(1-\rho_{ij})^2} \\
&= \frac{\rho_{ij}^2 \left[1-m\rho_{ij}^{m-1}+(m-1)\rho_{ij}^m\right]}{\left(1-\rho_{ij}\right)\left(1-\rho_{ij}^{m+1}\right)} \\
&= \frac{\rho_{ij}}{1-\rho_{ij}} - \frac{m\rho_{ij}^{m+1}+\rho_{ij}}{1-\rho_{ij}^{m+1}}
\end{aligned} \tag{5}$$

We can achieve the expectation of processed messages number $L_{\mathrm{ser}_{ij}}$ in (6), because the number of messages which are being processed is 0 (when CC_{ij} is idle) or 1 (when CC_{ij} is active). The expectation of message numbers in CC_{ij} is (7). The average number of messages arriving at CC_{ij} per unit time is (8).

$$\mathbf{L}_{\mathrm{ser}_{ij}} = 1 \times (1-p_0) + 0 \times p_0 = \frac{\rho_{ij}-\rho_{ij}^{m+1}}{1-\rho_{ij}^{m+1}} \tag{6}$$

$$\begin{aligned}
\mathbf{L}_{s_{ij}} &= L_{q_{ij}} + L_{\mathrm{ser}_{ij}} \\
&= L_{q_{ij}} + \frac{\rho_{ij}-\rho_{ij}^{m+1}}{1-\rho_{ij}^{m+1}} \\
&= \frac{\rho_{ij}-(m+1)\rho_{ij}^{m+1}+m\rho_{ij}^{m+2}}{(1-\rho_{ij})\left(1-\rho_{ij}^{m+1}\right)} \\
&= \frac{\rho_{ij}}{1-\rho_{ij}} - \frac{(m+1)\rho_{ij}^{m+1}}{1-\rho_{ij}^{m+1}}
\end{aligned} \tag{7}$$

$$\boldsymbol{\lambda}_{e_{ij}} = \lambda_{ij}(1-p_m) \tag{8}$$

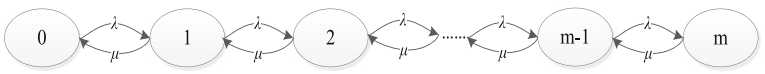

Fig. 3 $\mathrm{Load}_{\mathrm{notice}}$ signaling format

Based on Litter's Law and (5), (7), and (8), we get average waiting time for messages in (9) and average staying time (which consists of average waiting time and processed time) for messages in (10).

$$\begin{aligned}\mathbf{W}_{q_{ij}} &= \frac{L_{q_{ij}}}{\lambda_{e_{ij}}} \\ &= \frac{\rho_{ij}\left[1-m\rho_{ij}^{m-1}+(m-1)\rho_{ij}^{m}\right]}{\mu_{ij}(1-\rho_{ij})\left(1-\rho_{ij}^{m}\right)} \\ &= \frac{\rho_{ij}}{\mu_{ij}\left(1-\rho_{ij}\right)}-\frac{m\rho_{ij}^{m}}{\mu_{ij}\left(1-\rho_{ij}^{m}\right)}\end{aligned} \tag{9}$$

$$\begin{aligned}\mathbf{W}_{s_{ij}} &= \frac{L_{s_{ij}}}{\lambda_{e_{ij}}} \\ &= \frac{1-(m+1)\rho_{ij}^{m}+m\rho_{ij}^{m+1}}{\mu_{ij}\left(1-\rho_{ij}\right)\left(1-\rho_{ij}^{m}\right)} \\ &= \frac{1}{\mu_{ij}(1-\rho_{ij})}-\frac{m\rho_{ij}^{m}}{\mu_{ij}\left(1-\rho_{ij}^{m}\right)}\end{aligned} \tag{10}$$

3 Multi-controller sleeping management algorithm

In this section, we present the energy consumption management algorithm in HybridFlow and formulate this issue as *0-1* integer linear programming model, then utilize the *M-N* policy sleeping control in the system.

3.1 Problem formulation

The minimization problem of energy consumption can be formulated as *0-1* integer linear programming problem as follows:

$$\begin{aligned}\mathbf{min}tr(P*S') = \min\big(&p_{11}s_{11}+p_{12}s_{12}+\ldots+p_{1y}s_{1y} \\ &+p_{21}s_{21}+\ldots+p_{xy}s_{xy}\big)\end{aligned} \tag{11}$$

subject to

$$\begin{aligned}&\mathbf{tr}(\mathrm{MA}*S') \geq \mathrm{sum}(C) \\ &\mathbf{tr}(\mathrm{MA}*S') \geq \mathrm{sum}(L_q)\end{aligned} \tag{12}$$

where

1. MA: it is largest queue length matrix, in which MA_{ij} is the largest queue length in CC_{ij} and after queue length surpasses MA_{ij}, the extra messages will be discarded.
2. C: it is current queuing length matrix, in which c_{ij} is queuing length of CC_{ij} currently.
3. P: it is energy consumption matrix, in which p_{ij} is the energy consumption value of CC_{ij} at this moment which can be calculated with formula (1) and matrix MA, C. For example, p_{ij} calculates as follow:

$$\mathbf{P}_{ij} = P_{\mathrm{idle}} + (P_{\max} - P_{\mathrm{idle}})\left(2\frac{c_{ij}}{\mathrm{MA}_{ij}} - \left(\frac{c_{ij}}{\mathrm{MA}_{ij}}\right)^{2}\right) \tag{13}$$

4. S: it is switch matrix, in which s_{ij} takes the value of 1 if the CC_{ij} is active, and 0 otherwise.
5. L_q: it is average queuing length matrix, of which the corresponding element is the average queuing length in CC_{ij}, calculated by (5).
6. $tr()$: it is the trace of matrix.

In the above optimization, the first constraint represents after closing a part of cluster controllers; other clusters enable to accommodate the assigned load. The second constraint ensures system would not discard load because of the limitation of system capacity.

3.2 Algorithm procedure

In this section, we describe the procedure of energy consumption management algorithm in detail. The system is started. All cluster controllers report their current load c_{ij} to super controller every T seconds, namely the number of packet-in messages queuing in each cluster controller, using $\mathrm{load}_{\mathrm{notice}}$ signaling whose format is shown in Fig. 4.

When receiving $\mathrm{load}_{\mathrm{notice}}$ signaling, super controller puts c_{ij} into matrix C correspondingly and does the following judgment:

$$\begin{cases}\mathrm{sum}(C) > M*\mathrm{sum}(\mathrm{MA}), & \text{situation 1,}\\ \mathrm{sum}(C) \leq M*\mathrm{sum}(\mathrm{MA}), & \text{situation 2.}\end{cases} \tag{14}$$

where C is current queuing length matrix, MA is largest queue length matrix, and M is the starting threshold in *M-N* policy sleeping management algorithm. When situation 1 happens, the below algorithm will not be carried out. Cluster controllers wait for the next cycle to report $\mathrm{load}_{\mathrm{notice}}$ signaling. When situation 2 happens, it shows that the system is in the light traffic condition and there is no need to make all cluster controllers opened. Closing some cluster controllers enables to reduce the energy consumption of system. From (14), we can see that when M becomes larger, the threshold of starting the algorithm is bigger; the possibility of starting the algorithm is bigger, then the possibility of closing controllers is bigger; more controllers are to be closed in the system. In this case,

Fig. 4 State-flow Markov chain

energy consumption declines, but when more controllers are in sleeping mode, time delay of transferring sleeping controllers' packet-in messages increase. In a word, when M becomes bigger, energy consumption decreases and time delay increases. On the contrary, energy consumption increases and time delay decreases. We can conclude that there is fundamental tradeoff between energy consumption and time delay with a different M. The simulation which is shown in Section 4 also verifies the accuracy of the above analysis.

Energy consumption management algorithm is triggered. Using (11) and (12), the super controller knows how many cluster controllers needs to be opened and counts its number in (15). Which cluster controllers need to be closed has $C_{x*y}^{N_{\text{open}}}$ (which means combinatorial number) kinds of possibilities.

$$\mathbf{N}_{\text{open}} = \text{sum}(S) \tag{15}$$

In each scheme, super controller calculates the value of cost function in (16) and selects the scheme with minimum cost function.

$$\mathbf{cost} = \frac{T_{\text{delay}}}{E_{\text{save}}} \tag{16}$$

where T_{delay} is time delay of transferring messages of going to sleep controllers, which consists of T_{one} (the time delay of transferring messages in one cluster) and T_{across} (the time delay of transferring messages across clusters) in (17). E_{save} indicates saved energy calculated with (18).

$$\begin{aligned} \mathbf{T}_{\text{delay}} &= T_{\text{one}} + T_{\text{across}} \\ \mathbf{T}_{\text{one}} &= t_{\text{one}} * \text{Packet}_{\text{in}_{\text{one}}} \\ \mathbf{T}_{\text{across}} &= t_{\text{across}} * \text{Packet}_{\text{in}_{\text{across}}} \end{aligned} \tag{17}$$

$$\mathbf{E}_{\text{save}} = (P_{\max} - P_{\text{idle}}) \left[2 \left(\frac{c_{ij}}{\text{MA}_{ij}} \right) - \left(\frac{c_{ij}}{\text{MA}_{ij}} \right)^2 \right] \tag{18}$$

where t_{one} and t_{across} can achieve by testing experiment. $\text{Packet}_{\text{in}_{\text{one}}}$ is the number of messages transferring in one cluster, and $\text{Packet}_{\text{in}_{\text{across}}}$ is the number of messages transferring across clusters, which is calculated as follows.

The principle of messages transferring needs to be shown clearly.

1. Priority to transferring messages within one cluster.
2. Messages which are more than the capacity of this cluster are transferred across the cluster under the control of super controller.
3. System distributes loads according to the method of polling least-connection algorithm.

For example, assuming we calculate cost function of closing CC_{ij}, based on the first principle, firstly, super controller calculates L_{sc} in (19).

$$\mathbf{L}_{sc} = \left(\sum_{k \neq j} (\text{MA}_{ik} - c_{ik}) \right) - c_{ij} \tag{19}$$

If $L_{sc} \geq 0$, it shows that other cluster controllers in cluster i enable to accommodate load from CC_{ij}. According to the first principle, all of load of CC_{ij} are allocated within cluster i. Cluster controller CC_{ij} uses polling least-connection algorithm to determine the number of load distribution to other cluster controllers in the same cluster which is called $\text{Packet}_{\text{in}_{\text{one}}}$. Firstly, cluster controller CC_{ij} dispenses one message to the cluster controller which has least messages at this moment; this operation leads to the change of load distribution of the cluster. Secondly, cluster controller CC_{ij} also dispenses one message to least message controller. In this way, extra messages from cluster controller CC_{ij} enable to load balance between active cluster controller. The polling least-connection algorithm is as follows.

Algorithm 1 Polling least-connection algorithm

Input:
The array of messages number of controllers which receive loads,R;
The number of messages which need to be distributed to others, P;
Output:
The array of messages number of controllers which receive loads, R;

```
for i = 1; i < p; i++ do
    selecting the minimum value in R called min_R;
    min_R++;
end for
```

If $L_{sc} < 0$, based on the first and second principle, except for load that cluster i can accommodate, a part of messages from CC_{ij} should be transferred across cluster. The number of messages transferring in one cluster $\text{Packet}_{\text{in}_{\text{one}}}$ and the number of messages transferring across clusters $\text{Packet}_{\text{in}_{\text{across}}}$ calculate by super controller based on polling least-connection algorithm.

Using the above method, we can determine the number of transferring messages in one cluster and across clusters, then achieve T_{delay} by (17) and E_{save} by (18). Based on T_{delay} and E_{save}, the super controller calculates cost functions of each scheme by (16) and selects the scheme of minimum cost function, generates load transferring strategy, then sends sleeping messages and load transferring strategy to the selected cluster controllers. The cluster controllers which receive sleeping messages transfer all loads based on load transferring strategy from super controller and go to sleep.

Sleeping controllers detect the number of queuing length. When queuing length surpasses threshold N which is the threshold of setting up sleeping controllers, sleeping controllers awaken themselves and continue to deal with messages. Otherwise, cluster controllers are still in sleeping mode. When N becomes larger, the threshold of setting up sleeping controllers increases and sleeping controllers have a longer sleeping time, which reduces energy consumption. But in this case, packet-in messages queuing in the sleeping controllers will wait for a longer time, which gives rise to more time delay. In a word, when N becomes bigger, energy consumption decreases and time delay decreases. On the contrary, energy consumption increases and time delay decreases. We can conclude that there is also fundamental tradeoff between energy consumption and time delay with a different N. The simulation shown in Section 4 also verifies the accuracy of the above analysis. The algorithm and the actions performed on each of its steps are shown as follows.

Algorithm 2 Multi-controller sleeping management algorithm

1: The number of times system executing algorithm 2, k_0;
2: **for** $k = 1 : k_0$ **do**
3: cluster controllers report $load_{notice}$ signaling every T seconds;
4: **if** $sum(C) > M * sum(MA)$ **then**
5: return to 3;
6: **else**
7: Super controller calculates minimum of active controller based on (11) and (12);
8: Super controller calculates cost function of each scheme based on (16), (17) and (18), then selects the minimum scheme of them;
9: Super controller send load transferring strategy and sleeping messages to selected cluster controllers;
10: Selected cluster controllers transfer their loads;
11: Selected cluster controllers go to sleep and detect the number of queuing length
12: **if** number of queuing length in sleeping controller $< N$ **then**
13: return 11;
14: **else**
15: awaken themselves;
16: **end if**
17: **end if**
18: **end for**

4 Simulation results

In this section, we use computer simulation to evaluate the performance of energy consumption management algorithm. We first describe the simulation settings, then present the simulation results.

4.1 Simulation settings

1. *Simulation tools*: we simulate the system and algorithm in MATLAB.
2. *Network topologies*: the simulation is carried out in the topology as follows: there are four cluster controllers and one super controller in HybridFlow, which is to, $m = 2, n = 2$, respectively labeled in $CC_{11}, CC_{12}, CC_{21}, CC_{22}$. We assume the simulation is event-based, and when the network works, cluster controllers receive packet-in messages with $M/M/1/m$ queuing theory.
3. *Parameter settings*: the different seeds are employed in the simulation, and performances are averaged to estimate the performance of scheme. The values for all parameters in the simulation are summarized in Table 1.

4.2 Performance evaluation results

Figure 5 shows the total energy consumption of M-N policy sleeping mode, N policy sleeping mode and no sleeping mode with different average arriving rates. The other parameters are shown in Table 2, and the largest queue length in each cluster controller is the same, which is 20. In Fig. 5, with the increase of arriving rates, the total energy consumption is growing. The reason is that, as the arriving rates is increasing, more Packet-In messages accumulate in the cluster controllers, which results in the larger energy consumption due to load rising. In any case of different arriving rates, we can see in Fig. 5 that the energy consumption of HybridFlow with M-N policy sleeping mode is smaller than N policy sleeping mode and without sleeping mode, which shows the validity of

Table 1 Parameters setting in the simulation

Parameter	Value	Description
μ	30 μs	Serving rate at all of cluster controllers
Simulation time	10 s	The time of simulation
P_{max}	10 J	The energy consumption when there is full load
P_{idle}	0.05 J	The energy consumption when there is no load
t_{one}	0.02 ms	The time of transferring one message in one cluster
t_{across}	1 ms	The time of transferring one message across clusters
T	1 s	The updating cycle for $load_{notice}$ signaling
N	10	The threshold of awakening sleeping controllers
M	0.5	The threshold of executing the algorithm

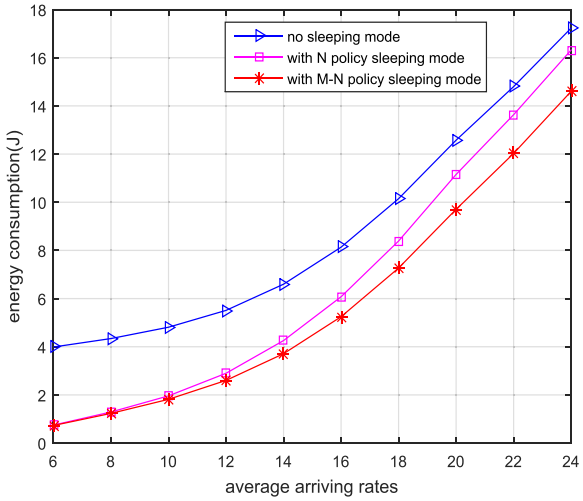

Fig. 5 Total energy consumption versus average arriving rates of *M-N* policy sleeping mode, *N* policy sleeping mode, and no sleeping mode

energy consumption management algorithm proposed in this paper. At the beginning of curves of with *M-N* policy sleeping mode and with *N* policy sleeping mode, their performance is similar, because when arriving rate is smaller, cluster controllers are often idle, in which case, *M-N* policy sleeping mode's advantage is not outstanding. As the increase of arriving rates, its performance is prominent gradually.

Figure 6 shows the relationship between average energy consumption in a packet and average delay in a packet of $M - N$ policy sleeping mode, N policy sleeping mode and no sleeping mode with different average λ. The other parameters are shown in Table 2, and the largest queue length in each cluster controller is the same, which is 20. In Fig. 6, it shows that the *M-N* policy sleeping mode and *N* policy sleeping mode both lead to the reduction of energy consumption but also introduce a longer time delay. Due to load transferring in the same cluster and between different clusters, our proposed mode introduces more time delay, but improves energy efficiency meanwhile.

Figure 7 shows the relationship between average energy in a packet and average delay in a packet in different M with the 25/ms of average arriving rate. We can observe that larger M introduces more time delay in a packet, but reduces average energy consumption in a packet. When M gets larger, the threshold of starting the algorithm is bigger; when the possibility of starting the algorithm is bigger, then the possibility of closing controllers is bigger; more controllers are to be closed in the system. Hence, energy consumption decreases. When more controllers are in sleeping mode, time delay of transferring sleeping controllers' packet-in messages increases. In a nutshell, when M becomes bigger, energy consumption decreases and time delay increases. On the contrary, energy consumption increases and time delay decreases. We can also conclude that there is the fundamental tradeoff between energy consumption and time delay with a different M.As M is smaller, performance has an obvious change, which is because the opening threshold in (14) will change a lot with changing of M. Once M is larger than a certain degree, and there is no more messages accumulating due to the given average arriving rate, performance will be stable. We can also conclude that there is fundamental tradeoff between energy consumption and time delay with a different M.

Figure 8 shows the relationship between the average energy consumption in a packet and average delay in a

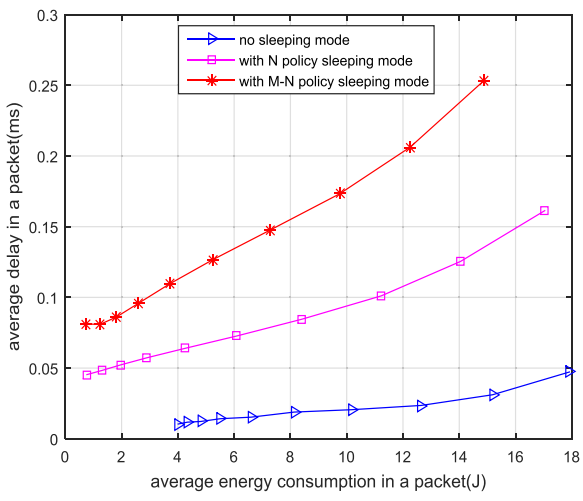

Fig. 6 Average energy consumption in a packet versus average delay in a packet in different arriving rates of *M-N* policy sleeping mode, *N* policy sleeping mode, and no sleeping mode

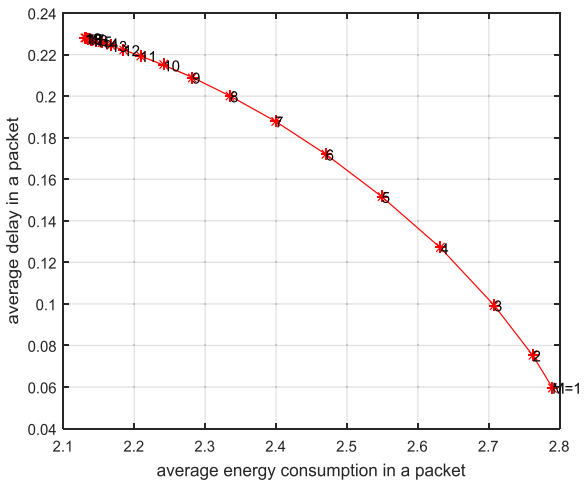

Fig. 7 Average energy consumption in a packet versus average delay in a packet with a different *M*

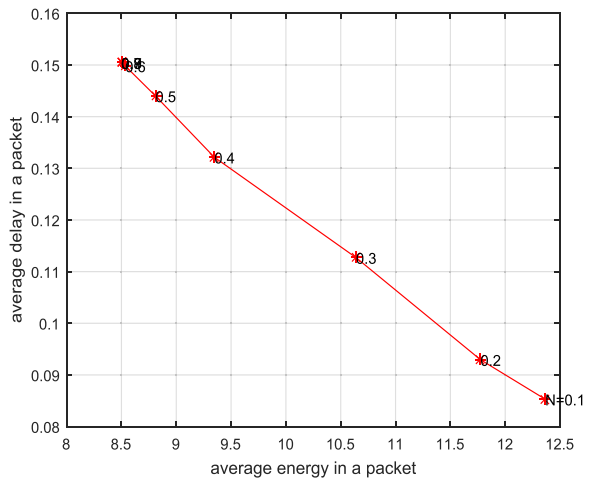

Fig. 8 Average energy consumption in a packet versus average delay in a packet with a different *N*

packet of different N with the 25/ms of average arriving rate. We can see that larger N can reduce average energy consumption in a packet. The reason is that, when N increases, the threshold of setting up sleeping controllers is raised, and controllers will have a longer sleeping time, which reduces average energy consumption. With the increase of N, the delay is growing. The reason is that, as the N is increasing, packet-in messages queuing in the sleeping controllers wait for a longer time, which gives rise to more time delay. Since larger N always reduces average energy consumption, but inevitably increases the average delay, we can conclude that there is the fundamental tradeoff between energy and delay with a different N.

5 Conclusions

In this paper, we have studied the method of switching off cluster controllers based on HybridFlow architecture to improve energy efficiency. First, we use queuing theory to model the operation procedure of controllers, formulate the energy consumption management issue as a *0-1* integer linear programming model. Through turning off the redundant controllers when the system is in the scenario of light traffic, the total energy consumption of the whole system can be cut down. Then, we present the processing of energy consumption management algorithm with *M-N* sleeping policy in details. Simulation results show that proposed algorithm exhibits better energy efficiency but introduces extra time delay. In further study, we need to improve energy efficiency, and reduce the time delay. Meanwhile, we need to find the relationship between energy efficiency, time delay and threshold M, N.

Acknowledgements
The project is supported by the Key Program of the National Natural Science Foundation of China (Grant No. 61431008).

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Information and Communication Engineering, Beijing University of Post and Telecommunications (BUPT), Xi Tu Cheng Road 10, 100876 Beijing, China. [2]China Mobile Group Design Institute Co., Ltd., Dä ling Street, 100080 Beijing, China.

Received: 20 December 2015 Accepted: 24 November 2016
Published online: 09 December 2016

References

1. Q Liang, X Cheng, SW Samn, NEW: network-e electronic warfare for target recognition. IEEE Trans. Aerospace Electron. Syst. **46**(2), 558–568 (2010). doi:10.1109/TAES.2010.5461641
2. Q Liang, X Cheng, KUPS: knowledge-based ubiquitous and persistent sensor networks for threat assessment. IEEE Trans. Aerospace Electron. Syst. **44**(3), 1060–1069 (2008). doi:10.1109/TAES.2008.4655363
3. Q Liang, X Cheng, SC Huang, D Chen, Opportunistic sensing in wireless sensor networks: theory and application. Comput. IEEE Trans. **63**(8), 2002–2010 (2014)
4. Q Liang, Automatic target recognition using waveform diversity in radar sensor networks. Pattern Recognit. Lett. **29**(3), 377–381 (2008)
5. Q Liang, in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*. Waveform design and diversity in radar sensor networks: theoretical analysis and application to automatic target recognition, vol. 2 (IEEE, 2006), pp. 684–689. doi:10.1109/SAHCN.2006.288531
6. W Xia, Y Wen, CH Foh, D Niyato, H Xie, A survey on software-defined networking. IEEE Commun. Surv. Tutor. **17**(1), 27–51 (2015). doi:10.1109/COMST.2014.2330903
7. BAA Nunes, M Mendonca, X-N Nguyen, K Obraczka, T Turletti, A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014). doi:10.1109/SURV.2014.012214.00180
8. N McKeown, T Anderson, H Balakrishnan, G Parulkar, L Peterson, J Rexford, S Shenker, J Turner, OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
9. AR Curtis, JC Mogul, J Tourrilhes, P Yalagandula, P Sharma, S Banerjee, DevoFlow: Scaling Flow Management for High-performance Networks. SIGCOMM Comput. Commun. Rev. **41**(4), 254–265 (2011). doi:10.1145/2043164.2018466. http://doi.acm.org/10.1145/2043164.2018466
10. F Yonghong, B Jun, W Jianping, C Ze, W Ke, L Min, A dormant multi-controller model for software defined networking. Commun. China. **11**(3), 45–55 (2014)
11. E Ng, Maestro: A system for scalable OpenFlow control. Technical report, TSEN Maestro-Technical Report TR10-08, Rice University (2010)
12. M Casado, MJ Freedman, J Pettit, J Luo, N McKeown, S Shenker, Ethane: Taking Control of the Enterprise. SIGCOMM Comput. Commun. Rev. **37**(4), 1–12 (2007). doi:10.1145/1282427.1282382. http://doi.acm.org/10.1145/1282427.1282382
13. Y Hu, W Wang, X Gong, X Que, S Cheng, in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*. BalanceFlow: Controller load balancing for OpenFlow networks, vol. 2 (IEEE, 2012), pp. 780–785. doi:10.1109/CCIS.2012.6664282
14. A Tootoonchian, Y Ganjali, in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. HyperFlow: A distributed control plane for OpenFlow, (2010), pp. 3–3
15. T Koponen, M Casado, N Gude, J Stribling, L Poutievski, M Zhu, R Ramanathan, Y Iwata, H Inoue, T Hama, S Shenker, in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. Onix: d distributed control platform for large-scale production networks (USENIX Association, Berkeley, 2010), pp. 351–364. http://dl.acm.org/citation.cfm?id=1924943.1924968
16. G Warkozek, E Drayer, V Debusschere, S Bacha, in *2012 IEEE International Conference on Industrial Technology*. A new approach to model energy consumption of servers in data centers (IEEE, 2012), pp. 211–216. doi:10.1109/ICIT.2012.6209940