future internet

Article

Dynamic Traffic Scheduling and Congestion Control across Data Centers Based on SDN

Dong Sun [1], Kaixin Zhao [2], Yaming Fang [3] and Jie Cui [3,*]

1 Experimental Management Center, Henan Institute of Technology, Xinxiang 453000, China; dongdong@hait.edu.cn
2 Department of Computer Science and Technology, Henan Institute of Technology, Xinxiang 453003, China; 13849371368@hait.edu.cn
3 School of Computer Science and Technology, Anhui University, Hefei 230039, China; 13721032191@139.com
* Corresponding: cuijie@mail.ustc.edu.cn

Received: 20 June 2018; Accepted: 7 July 2018; Published: 9 July 2018

Abstract: Software-defined Networking (SDN) and Data Center Network (DCN) are receiving considerable attention and eliciting widespread interest from both academia and industry. When the traditionally shortest path routing protocol among multiple data centers is used, congestion will frequently occur in the shortest path link, which may severely reduce the quality of network services due to long delay and low throughput. The flexibility and agility of SDN can effectively ameliorate the aforementioned problem. However, the utilization of link resources across data centers is still insufficient, and has not yet been well addressed. In this paper, we focused on this issue and proposed an intelligent approach of real-time processing and dynamic scheduling that could make full use of the network resources. The traffic among the data centers could be classified into different types, and different strategies were proposed for these types of real-time traffic. Considering the prolonged occupation of the bandwidth by malicious flows, we employed the multilevel feedback queue mechanism and proposed an effective congestion control algorithm. Simulation experiments showed that our scheme exhibited the favorable feasibility and demonstrated a better traffic scheduling effect and great improvement in bandwidth utilization across data centers.

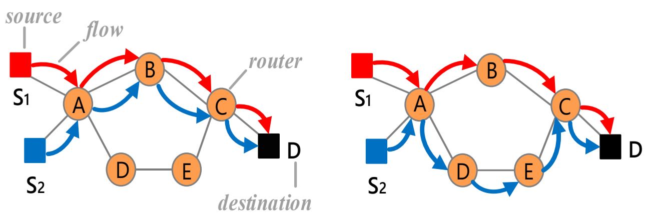
Keywords: data center; dynamic scheduling; congestion control; OpenFlow; software-defined networking

1. Introduction

Big data has become one of the hottest topics among academia and industry. With the development of big data, the amount of data from different sources such as the Internet of Things, social networking websites, and scientific research is increasing at an exponential rate [1]. The scale of data centers has gradually extended. Meanwhile, an increasing number of data are being transmitted in data center networks, and traffic exchange among the servers in data centers has also been growing fast. The most direct result may be low utilization ratio, congestion problems, service latency, and even DDOS attacks [2]. Data centers are always interconnected through wide area networks [3]. When traditional routing protocols are used in data center networks, flows are forced to preempt the shortest path to be routed and forwarded, which might lead the shortest path link to be under full load while some new flows are still competing for it, and other links are under low load. Without shunting for flows, the link would easily be congested and unable to provide normal network services. The most direct, but expensive solution, is to remold and upgrade the network. However, on account of the scale, complexity, and heterogeneity of current computer networks, traditional approaches to configuring the network devices, monitoring and optimizing network performance, identifying and solving network problems, and planning network growth would become nearly impossible and inefficient [4].

Software-defined networking (SDN) is one of the most notable forms of computer networking. It is receiving considerable attention from academic researchers, industry researchers, network operators, and some large and medium-sized networking enterprises [5]. SDN is considered as a promising modality to rearchitect our traditional network [6]. The core idea of SDN is to decouple the control plane from the data plane to achieve flexible network management, efficient network operation, and low-cost maintenance through software programming [7]. Specifically, infrastructure devices merely execute packet forwarding depending on the rules installed by the SDN controller [8]. In the control plane, the SDN controller can oversee the topology of the underlying network and provide an agile and efficient platform for the application plane to implement various network services. In this new network paradigm, innovative solutions for realizing specific and flexible services can be implemented quickly and efficiently in the form of software and deployed in real networks with real-time traffic. In addition, this paradigm allows for the logically centralized control and management of network devices in the data plane in accordance with a global and simultaneous network view and real-time network information. Compared with traditional networks, it is much easier to develop and deploy applications in SDN [9].

This paper concentrated on the problem of data center traffic management and attempted to avoid congestion to make full use of the bandwidth resources. We proposed a new solution based on SDN for traffic engineering in data center networks by developing two modules on top of an open source SDN controller called Floodlight. The traffic among the data centers can be classified into different types. Different strategies are adopted upon the real-time changes of the link states. The dynamic traffic scheduling can take full advantage of the network resources better. Given the prolonged occupation of the bandwidth by malicious flows, we proposed an effective congestion control algorithm by adopting a multilevel feedback queue mechanism. Considering a large-scale IP network with multiple data centers, the basic components are shown in Figure 1a. With traditional routing protocol, all flows from S_1 and S_2 to D would traverse or congest paths (A, B, C) without using (A, D, E, C). In contrast, the scheme DSCSD in this paper can route flows from S_1 and S_2 to D to select different paths according to their types and the real-time link information (Figure 1b), and the congestion can also be well controlled.



(**a**) (**b**)

Figure 1. Flows that preempt the shortest path (**a**) can be shunted into two different paths (**b**) with DSCSD.

The main contributions of this paper are as follows. First, we proposed a new approach for traffic scheduling that could route a newly arrived flow based on the real-time link information, and also dynamically schedule flows on the link. Better than traditional approaches and the SDN-based scheme with threshold value, we could improve the efficiency of the link utilizing the shortest paths. Second, we innovatively adopted a multilevel feedback queue mechanism of congestion control suitable for different types of flows and could realize anomaly detection by preventing malicious flows from occupying the bandwidth for a long time.

The remainder of the paper is organized as follows: In Section 2, we review the traditional network and SDN used in data centers. Then, we elaborate on the design and implementation of our proposed scheme, DSCSD (Dynamic Scheduling and Congestion control across data centers based on SDN) in

Section 3. The performance evaluation of DSCSD is presented in Section 4. Finally, Section 5 concludes the paper.

2. Related Work

This section reviews recent studies on traffic engineering in data centers which served as our research background and theoretical foundation. We discussed this from two aspects including a data center based on a traditional network and a data center based on SDN as well as its challenges. The first aspect explains the development of distributed data centers and the limitations of the traditional network protocols used in current data centers. These limitations exactly prove the demand of our proposed research. The other aspect shows some recent research progress of SDN-based data centers.

2.1. Data Center Based on Traditional Network

In the traditional data center model, the traffic is mainly generated between the server and the client, with a low proportion of east–west traffic among the data centers [10]. With the extensive use of the Internet and the integration of the mobile network, Internet of Vehicle [11], cloud computing, big data, and other new generations of network technology and development have come into being to deal with massive-scale data and large-scale distributed data centers, bringing about the accelerated growth of east–west traffic exchanged among servers, for example, the Google File System (GFS) [12], Hadoop Distributed File System (HDFS) [13], and Google framework MapReduce [14]. However, when the traditionally shortest path routing protocols in current data centers are used, congestion will frequently occur in the shortest path link, and it may further reduce the quality of network services due to the long delay and low throughput.

Traffic scheduling and congestion control are important technologies to maintain network capacity and improve network efficiency. Traditional networks have some congenital defects. The main disadvantages can be summarized as follows: First, there is no global coordinative optimization. Each node independently implements the traffic control strategy, which can only achieve the local optimum. Moreover, there is no dynamic and self-adaptive adjustment. The predefined strategies in routers cannot meet the frequently changing demands of business flows. In addition, traditional networks find it difficult to achieve the effective and accurate control of every network device. The configurations of network devices are numerous and diverse where the commands are so complicated that it is very difficult to find the network errors caused by configurations. Consequently, it is of great urgency to figure out how to effectively manage and dominate network traffic, which has pushed network architects to take advantage of SDN to address these problems in data centers.

2.2. Data Center Based on SDN

A higher-level of visibility and a fine-grained control of the entire network can be achieved in the SDN paradigm. The SDN controller is able to program infrastructural forwarding devices in the data plane to monitor and manage network packets passing through these devices in a fine-grained way. Therefore, we can use the SDN controller to implement the periodic collection of these statistics. Furthermore, we can also obtain a centralized view of the network status for the SDN applications via open APIs and notify the up-level applications of a change in a real-time network [15].

Data centers are typically composed of thousands of high-speed links such as the 10 G Ethernet. In conventional packet capture mechanisms like switch port analyzer and port mirroring, infeasibility is completely reflected from a cost and scale perspective because a stupendous number of physical ports might be used. Adopting the SDN-based approaches to data center networks has become the focus of current research [1]. Tavakoli et al. [16] first applied SDN to data center networks by using NOX (the first SDN controller) to efficiently actualize addressing and routing of the typical data center networks VL2 and PortLand. Tootoonchian [17] proposed HyperFlow, a new distributed control plan for OpenFlow. Multiple controllers cooperate with each other to make up for the low scalability of a single controller while the advantage of centralized management is retained. The cooperation of distributed controllers can realize the expansion of the network and conveniently manage network devices [18–20]. Koponen et al. [21] presented Onix, on which the SDN control plane could be implemented as a distributed system. Benson et al. [22] proposed MicroTE, a fine-grained routing approach for data centers where he found a lack of multi-path routing, plus an overall view of workloads and routing strategies in data center networks based on a traditional network. Hindman et al. [23] presented Mesos, a platform for fine-grained resource sharing in data centers. Curtis [24] proposed Mahout to handle elephant flows (i.e., large flows) while large flows in Socket buffers were detected. Due to the burst of the traffic and the failure of the equipment, congestion and failure frequently occur. Large flows and small flows are always mingled in data center networks. Kanagavelu et al. [25] proposed a flow-based edge–to–edge rerouting scheme. When congestions appeared, it focused on rerouting the large flows to alternative links. The reason for this mechanism is that shifting these short-lived small flows among links would additionally increase the overhead and latency. Khurshid et al. [26] provided a layer between a software-defined networking controller and network devices called VeriFlow. It is a network debugging tool to find faulty rules issued by SDN applications and anomalous network behavior. FP Tso [27] presented the Baatdaat flow scheduling algorithm using spare data center network capacity to mitigate the performance degradation of heavily utilized links. This could ensure real-time dynamic scheduling, which could avoid congestion caused by instantaneous flows. Li et al. [28] proposed a traffic scheduling solution based on the fuzzy synthetic evaluation mechanism, and the path could be dynamically adjusted according to the overall view of the network.

In summary, most of the current active approaches solving traffic engineering in data center networks based on SDN have two alternatives. The first common approach is based on a single centralized controller, with the problem of scalability and reliability. The other method is based on multiple controllers, but the cooperation of multiple controllers and the consistent update mechanism has not been well addressed thus far [29]. Our proposed scheme is an attempt that first applies the multilevel feedback queue mechanism to traffic scheduling and congestion control across data centers based on SDN, and can also provide a new solution to congestion caused by the prolonged occupation of malicious flows.

3. The Design and Implementation of DSCSD

Usually, the links among data centers are highly probable for reuse, and the burst of instantaneous traffic leads to the instability and dynamicity of the network. Traffic scheduling and congestion control among multiple data centers can be well addressed with DSCSD by taking advantage of the flexibility and agility of SDN. Data centers are always interconnected through wide area networks. The network traffic is not constant at all times and has an unbalanced distribution. The traffic during the peak hours could increase by twice that of the average link load. However, the traditional routing protocols route and forward flows abide by the shortest paths algorithm, without shunting away flows to balance the link load. To satisfy the requirement of bandwidth during peak hours, we have no choice but to purchase 2–3$\times$ bandwidth as well as upgrade to large-scale routing equipment. As a result, the cost of network operation, administration, and maintenance are increased sharply, and also causes the waste of bandwidth in ordinary times. If we can meet the requirements of network services, the average link utilization rate of wide area networks can only achieve 30–40% [30]. Saying that some malicious flows extend the occupation of links, congestion might bring about interruptions in the transmission of their services [31–34]. In this paper, we propose DSCSD, a scheme of dynamic traffic scheduling and congestion control with a multilevel feedback queue mechanism that can address the problems above-mentioned to some extent. The following sections explain the design in detail and its implementation process.

3.1. System Model

Our scheme was designed and innovated on the basis of data center networking. In the following presentation, we briefly describe the components of the system model in Figure 2. (i) Data center servers. PC1, PC2, and PC3 are three individual data centers, and the fs from PC1 to PC3 takes priority over the flows from PC2 to PC3. (ii) OpenFlow switches. With the five switches, two distinct paths are formed, and the path (S1, S4, S5) is the shortest path leaving (S1, S2, S3, S4) the second shortest. (iii) SDN controller. We selected the open source OpenFlow controller Floodlight as the centralized controller and it can be used to control the behavior of OpenFlow switches by adding, updating, and deleting flow table entries in the switches.

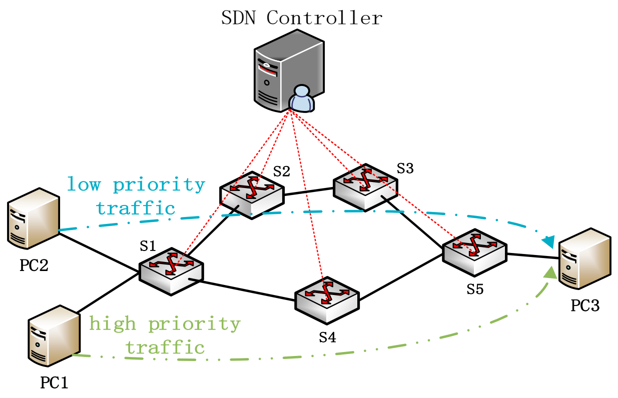


Figure 2. System model.

3.2. Dynamic Traffic Scheduling

We classified the flows among multiple data centers into different types. As for data duplication flows, we set a lower priority for them, while the other flows of high quality requirement had a higher priority. The higher priority flows traverse the shortest path, and the lower priority can select a path depending on the real-time link states. The concrete link states on the shortest path are categorized into the following four cases.

State 1: The time when a low priority flow arrives. In this state, we still have several different possibilities. First, if bandwidth remains on the shortest path (S1, S4, S5), then it will directly select this path. If the shortest path (S1, S4, S5) is fully occupied by a high priority flow, then it will have no choice but be transmitted on the second shortest path (S1, S2, S3, S5). The last possibility is that this flow should go into the congestion control with multilevel feedback queues, given all paths have no bandwidth to be used.

State 2: The time when a high priority flow arrives. In this state, we also have several different possibilities. First, if bandwidth remains on the shortest path (S1, S4, S5), then it will directly select this path. If the shortest path (S1, S4, S5) is fully occupied by a high priority flow, then it will have no choice but be transmitted on the second shortest path (S1, S2, S3, S5). The last possibility is that this flow should go into the congestion control with multilevel feedback queues, given all paths have no bandwidth to be used.

State 3: The time when a high priority flow is transmitting on the link. Due to the high priority, we do nothing with it.

State 4: The time when a low priority flow is transmitting on the link. From state 1, we can learn that the low priority flow can select any one of the two paths adapting to data fluctuation. Therefore, if there is no available bandwidth on the shortest path link when a new flow with high priority arrives at this point, it will vacate the shortest path link and be scheduled to the second shortest path link. Otherwise, if the new flow has an equally low priority and no subsequent low priority flow needs transmitting, the shortest path is allotted to the newly arrived flow.

According to the aforementioned analysis of the link states, we can realize dynamic traffic scheduling to improve the utilization of bandwidth resources.

3.3. Congestion Control with Multilevel Feedback Queues

As part of DSCSD, dynamic traffic scheduling has an obvious effect on the classification and diversion of flow. However, when malicious flows exist, congestion also might occur in the shortest path link due to the long-term occupation of limited link resources and the burst of network instantaneous traffic. The algorithm of congestion control with multilevel feedback queues not only provides a queuing service for congested flows, but also settles the problem of vicious prolonged occupation. The specific description is shown as follows.

In the initial phase of the scheme, we define two multilevel feedback queues to store the flows waiting to be scheduled. One is the multilevel feedback queue of low priority and the other is the multilevel feedback queue of high priority. Then, in each feedback queue, we define three sub-queues, and give the highest priority to flow waiting queue 1, the second highest priority to flow waiting queue 2, and the lowest priority to flow waiting queue 3. As is shown in Figure 3, the transmission time of these flow waiting queues are different after being scheduled. Sub-queue 1 has the time t, sub-queue 2 has the time 2t that is twice the time of sub-queue 1, and sub-queue 3 has the time 3t that is triple the time of sub-queue 1.

For the pseudo codes of the algorithm, see Algorithm 1.

Algorithm 1: The Algorithm of Multilevel Feedback Queue

```
Input:  G: topology of data center network;
    $F_{active}$: set of active flows;
        $F_{suspend}$: set of suspended flows;
        $F_{new}$: a new flow;
        $F_{\text{first}}$: a flow from the first of the $F_{suspend}$.
    Output: {<e.state, e.path>}: scheduling state and path selection of each flow in G. When a new
flow arrives
1  if (e.path = IDLEPATH) then
2     $F_{active} \leftarrow F_{active} + F_{new}$;
3  end
4  else $F_{suspend} \leftarrow F_{suspend} + F_{new}$;
5     while $\left(F_{suspend} \neq \varnothing\right)$ do
6     if (e.path = IDLEPATH) then
7        if $\left(F_{suspend1} \neq \varnothing\right)$ then
8           Select the flow at the first of $F_{suspend1}$;
9           $F_{suspend1} \leftarrow F_{suspend1} - F_{first}$;
10            $F_{active} \leftarrow F_{active} + F_{first}$;
11            Transmission time of the $F_{first}$ is t;
12         end
13         else if $\left(F_{suspend2} \neq \varnothing\right)$ then
14            Select the flow at the first of $F_{suspend2}$;
15            $F_{suspend2} \leftarrow F_{suspend2} - F_{first}$;
16            $F_{active} \leftarrow F_{active} + F_{first}$;
17            Transitssion time of the $F_{first}$ is 2t;
18         end
19      else
20            Select the flow at the first of $F_{suspend3}$;
21            $F_{suspend3} \leftarrow F_{suspend3} - F_{first}$;
22            $F_{active} \leftarrow F_{active} + F_{first}$;
23            Transit time of the $F_{first}$ is 3t;
24      end
25   return {<e.state, e.path>};
```

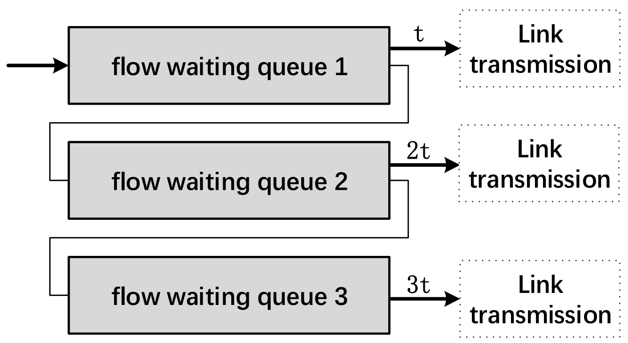


Figure 3. Multilevel feedback queue.

4. Experiment Results and Performance Analysis

The SDN controller selected in our scheme is a free open source Floodlight that runs in the eclipse environment on the Ubuntu system. The virtual switch uses open source Open vSwitch 2.3.0. The virtual network was created by Mininet in Ubuntu 14.04.

We implemented DSCSD on top of the Floodlight v1.2 controller in our simulation experiments. Floodlight is a free source, and modules can be randomly added and deleted, so that it provides much more convenience for our test. In our experimental setup, the virtual switch was created by Open vSwitch. The Floodlight was chosen as the SDN controller. We added two modules named initflowtable and trafficmanage on Floodlight. The module initflowable was used to initialize the variable with a value and install some flow tables of preprocessing. The main function of our scheme was the module trafficmanage that could monitor the Packet-in messages and the Flow-removed messages of the OpenFlow protocol and recode the installed flow tables as well as link information including congestion and bandwidth. When a new flow arrives or a flow needs to be scheduled, the module trafficmanage can achieve accurate and real-time detection and control.

As shown in Figure 4, we used several virtual hosts to simulate three data centers; h1 and h2 were the servers of two individual data centers; c1, c2, c3, c4, and c5 were collectively used as another data center. Certainly, we should assign virtual IP addresses and MAC addresses for them. We also used the flows from any one of h1 and h2 to any one of c1, c2, c3, c4, and c5 to simulate the interconnected flows among the data centers. In accordance with the system model in Figure 2, the path (S1, S4, S5) was the shortest path leaving (S1, S2, S3, S4) the second shortest. Here, we conducted two tests to verify the effectiveness and performance of DSCSD.

Test 1: Verify the effectiveness of DSCSD. We used Mininet to set a 4 M-bandwidth for both the shortest path and the second shortest path, and we also utilized the tool iperf to simulate four flows from h1 to c1 and c2, and from h2 to c3 and c4. Each link was equipped with the requirement of a 2 M-bandwidth. We discussed and analyzed the performance of the traditional network and DSCSD. Then, we chose a span under the selfsame condition to test the loss tolerance and real-time bandwidth of these flows. Some of these data are presented in Table 1. Here, we adopted four symbolic notations. The notation “h1–1” means the traffic from h1 to c1. By analogy, “h1–2” means the traffic from h1 to c2; “h2–c3” means the traffic from h2 to c3; and “h2–c4” means the traffic from h2 to c4. We also calculated the overall link utilization of both cases as is shown in Figure 5.

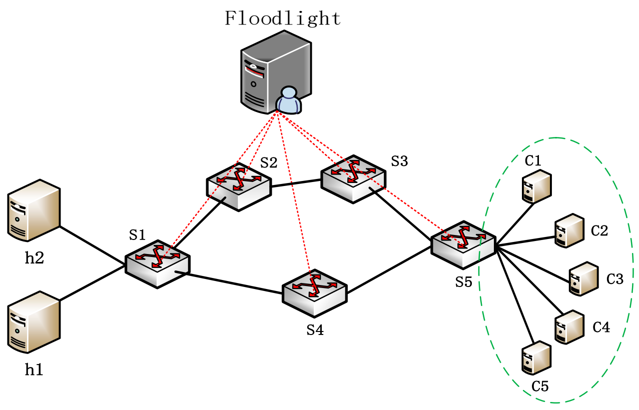


Figure 4. Experimental topology.

Table 1. Real-time bandwidth comparison.

bps	Traditional Network				DSCSD			
Time	**h1–1**	**h1–c2**	**h2–c3**	**h2–c4**	**h2–c4**	**h2–c4**	**h2–c4**	**h2–c4**
0–1 s	106 k	1.87 M	1.91 M	25.3 k	1.95 M	1.94 M	1.95 M	1.94 M
1–2 s	11.8 k	1.86 M	1.81 M	70.6 k	1.94 M	1.95 M	1.94 M	1.94 M
2–3 s	11.8 k	1.89 M	1.89 M	35.3 k	1.94 M	1.94 M	1.94 M	1.95 M
3–4 s	153 k	1.92 M	1.80 M	11.8 k	1.95 M	1.94 M	1.94 M	1.94 M
4–5 s	35.3 k	1.75 M	1.94 M	176 k	1.94 M	1.95 M	1.95 M	1.94 M
5–6 s	82.3 k	1.94 M	1.89 M	82.3 k	1.82 M	1.94 M	1.94 M	1.94 M
6–7 s	35.3 k	1.95 M	1.88 M	23.5 k	1.94 M	1.94 M	1.87 M	1.95 M
7–8 s	23.5 k	1.94 M	1.88 M	58.8 k	1.95 M	1.94 M	1.93 M	1.94 M
8–9 s	11.8 k	1.89 M	1.92 M	23.5 k	1.95 M	1.95 M	1.94 M	1.94 M
9–10 s	23.5 k	1.91 M	1.90 M	11.8 k	1.94 M	1.95 M	1.94 M	1.93 M

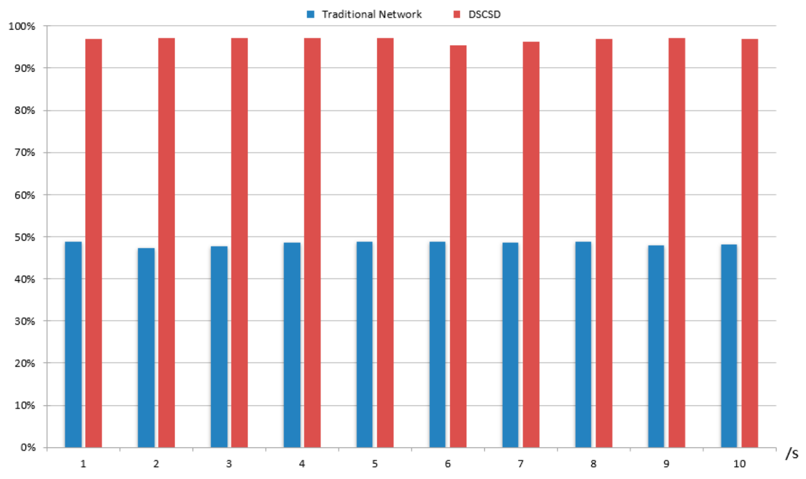


Figure 5. Overall link utilization comparison.

In Test 1, it held that the loss tolerance of the four flows in a traditional network was high, with two of them close to 100%. As for the overall link utilization, the average link utilization of our scheme was kept at 97%, while that of traditional network was just 48%. For a comparison, we verified the high improvement of link utilization by using DSCSD.

Test 2: Congestion control with multilevel feedback queues. In this test, we still used the simulation of Test 1. The difference was that we were only interested in the shortest path (S1, S4, S5) and set a 5 M-bandwidth. The time t was 20 s. Then, we let c5 send UDP flows to h1 with a 3 M-bandwidth consistently, while c1, c2, c3, and c4 sent UDP flows to h2 with a 1 M-bandwidth.

Here, we divided Test 2 into two cases: real-time bandwidth without congestion control queues and real-time bandwidth with congestion control queues, then we discussed and analyzed the effectiveness of congestion control in these two cases. The real-time bandwidths in two cases are respectively shown in Figures 6 and 7.

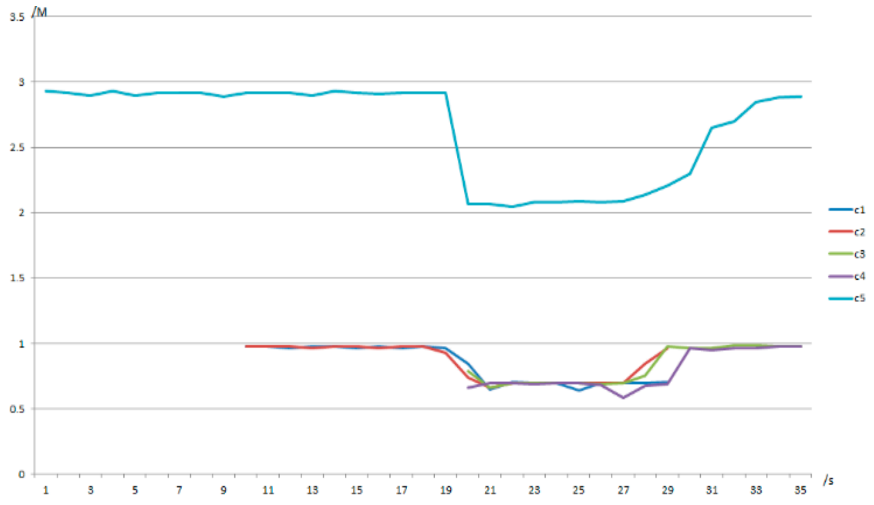


Figure 6. Real-time bandwidth without congestion control queues.

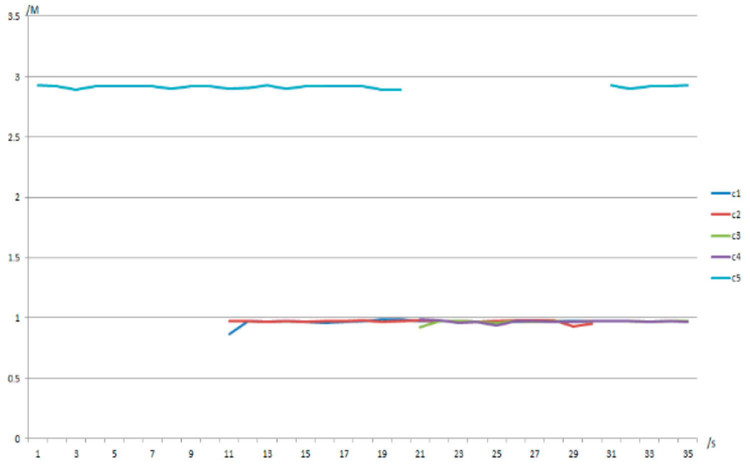


Figure 7. Real-time bandwidth with congestion control queues.

At 10 s, the path (S1, S4, S5) has been saturated with the flows from c5, c1, and c2. At 20 s, we received the request of the normal flows from c3 and c4. At this time, if we do not use congestion control queues (case 1), the flow from c5 with high priority directly affects the quality of the transmission of other flows, which can be observed in Figure 6. In contrast, we noticed the comparison in Figure 7. If we used congestion control queues (case 2), when we received the request of normal flows from c3 and c4, flows from c5 were scheduled into the congestion control queues. Considering the bandwidth remained on the path, the flow from c5 will be rescheduled to the path to be transmitted. With this mechanism, we can provide a new solution to control the congestion caused by the prolonged occupation of malicious flows.

As shown in Figure 8, we demonstrated the comparison of the link delay in the two aforementioned cases. We can easily draw the conclusion that by using the congestion control queue, the congestion caused by malicious flows can be well addressed with a relative reduction in link delay.

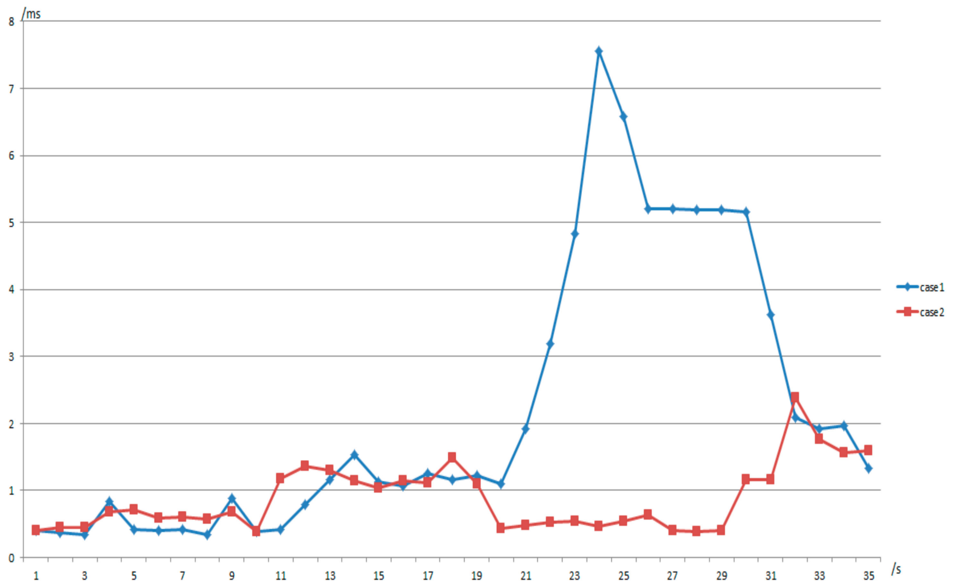


Figure 8. Comparison of the link delay.

5. Conclusions and Future Work

In this paper, we focused on the problem of traffic scheduling and congestion control across data centers and aimed to provide an approach that could greatly improve link utilization. To realize this goal, we designed DSCSD, a dynamic traffic scheduling and congestion control scheme across data centers based on SDN. The moment a flow arrives, it relies on the connection of traffic parameters and link information to select paths. Furthermore, it can achieve real-time dynamic scheduling to avoid congestion caused by the burst of instantaneous traffic, and can also balance the link loads. Compared with traditional approaches, the experiment and analysis had an obvious effect on the classification and diversion of flows, thereby improving the link utilization across data centers. Better than the SDN-based scheme with threshold value, the real-time monitoring and dynamic scheduling of the shortest paths were fully reflected. Meanwhile, we innovatively adopted the mechanism of a multilevel feedback queue for congestion control, which is suitable for different types of flows and can implement anomaly detection by preventing malicious flow from chronically occupying the bandwidth.

Our proposed DSCSD scheme can be easily deployed to the existing data center networks to deal with the low utilization of link resources such as the data centers of live video streaming platforms and online video platforms. Although our scheme solved the traffic scheduling problem efficiently, there are still some limitations. First, our scheme is based on SDN, but DSCSD is not applicable in traditional network environments or a hybrid network environment. Second, more fine-grained and flexible hierarchical control might be helpful to further enhance the experimental result. In addition, we did not take into account the issue of energy saving in the traffic scheduling and congestion control of data center networks. Accordingly, we would like to enrich DSCSD with energy management in the future.

Author Contributions: D.S. conceived and designed the system model and algorithm; K.Z. was responsible for literature retrieval and chart making; Y.F. performed the experiments, analyzed the data, and wrote the paper; J.C. designed the research plan, conceived the algorithm model, and polished the paper.

Funding: This research was funded by the [National Natural Science Foundation of China] grant number [61502008], the [Key Scientific Research Project of Henan Higher Education] grant number [16A520084], the [Natural Science Foundation of Anhui Province] grant number [1508085QF132] and [the Doctoral Research Start-Up Funds Project of Anhui University].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cui, L.; Yu, F.R.; Yan, Q. When big data meets software-defined networking: SDN for big data and big data for SDN. *IEEE Netw.* **2016**, *30*, 58–65. [CrossRef]
2. Lan, Y.L.; Wang, K.; Hsu, Y.H. Dynamic load-balanced path optimization in SDN-based data center networks. In Proceedings of the 10th International Symposium on Communication Systems, Networks and Digital Signal Processing, Prague, Czech Republic, 20–22 July 2016; pp. 1–6.
3. Ghaffarinejad, A.; Syrotiuk, V.R. Load Balancing in a Campus Network Using Software Defined Networking. In Proceedings of the Third GENI Research and Educational Experiment Workshop, Atlanta, GA, USA, 19–20 March 2014; pp. 75–76.
4. Xia, W.; Wen, Y.; Foh, C.; Niyato, D.; Xie, H. A Survey on Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 27–51. [CrossRef]
5. Nunes, A.; Mendonca, M.; Nguyen, X.; Obraczka, K.; Turletti, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1617–1634. [CrossRef]
6. Lin, P.; Bi, J.; Wang, Y. WEBridge: West–east bridge for distributed heterogeneous SDN NOSes peering. *Secur. Commun. Netw.* **2015**, *8*, 1926–1942. [CrossRef]
7. Sezer, S.; Scott-Hayward, S.; Chouhan, P.K.; Fraser, B.; Lake, D.; Finnegan, J.; Viljoen, N.; Miller, M.; Rao, N. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Commun. Mag.* **2013**, *51*, 36–43. [CrossRef]
8. Mckeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
9. Kim, H.; Feamster, N. Improving network management with software defined networking. *IEEE Commun. Mag.* **2013**, *51*, 114–119. [CrossRef]
10. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P. The cost of a cloud: Research problems in data center networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *39*, 68–73. [CrossRef]
11. Cheng, J.; Cheng, J.; Zhou, M.; Liu, F.; Gao, S.; Liu, C. Routing in Internet of Vehicles: A Review. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2339–2352. [CrossRef]
12. Ghemawat, S.; Gobioff, H.; Leung, S.T. The Google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 19–22 October 2003; pp. 29–43.
13. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The Hadoop Distributed File System. In Proceedings of the IEEE 26th Symposium on MASS Storage Systems and Technologies, Incline Village, NV, USA, 3–7 May 2010; pp. 1–10.
14. Dean, J.; Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. In Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation, San Francisco, CA, USA, 6–8 December 2004; pp. 137–150.
15. Ali, S.T.; Sivaraman, V.; Radford, A.; Jha, S. A Survey of Securing Networks Using Software Defined Networking. *IEEE Trans. Reliab.* **2015**, *64*, 1–12. [CrossRef]
16. Tavakoli, A.; Casado, M.; Koponen, T.; Shenker, S. Applying NO_X to the Datacenter. In Proceedings of the Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII), New York, NY, USA, 22–23 October 2009.
17. Tootoonchian, A.; Ganjali, Y. HyperFlow: A distributed control plane for OpenFlow. In Proceedings of the Internet Network Management Conference on Research on Enterprise Networking, San Jose, CA, USA, 27 April 2010; p. 3.
18. Yu, Y.; Lin, Y.; Zhang, J.; Zhao, Y.; Han, J.; Zheng, H.; Cui, Y.; Xiao, M.; Li, H.; Peng, Y.; et al. Field Demonstration of Datacenter Resource Migration via Multi-Domain Software Defined Transport Networks with Multi-Controller Collaboration. In Proceedings of the Optical Fiber Communication Conference, San Francisco, CA, USA, 9–13 March 2014; pp. 1–3.
19. Zhang, C.; Hu, J.; Qiu, J.; Chen, Q. Reliable Output Feedback Control for T-S Fuzzy Systems with Decentralized Event Triggering Communication and Actuator Failures. *IEEE Trans. Cybern.* **2017**, *47*, 2592–2602. [CrossRef] [PubMed]

20. Zhang, C.; Feng, G.; Qiu, J.; Zhang, W. T-S Fuzzy-model-based Piecewise H_infinity Output Feedback Controller Design for Networked Nonlinear Systems with Medium Access Constraint. *Fuzzy Sets Syst.* **2014**, *248*, 86–105. [CrossRef]
21. Koponen, T.; Casado, M.; Gude, N.S.; Stribling, J.; Poutievski, L.; Zhu, M.; Ramanathan, R.; Iwata, Y.; Inoue, H.; Hama, T.; et al. Onix: A distributed control platform for large-scale production networks. In Proceedings of the Usenix Symposium on Operating Systems Design and Implementation, Vancouver, BC, Canada, 4–6 October 2010; pp. 351–364.
22. Benson, T.; Anand, A.; Akella, A.; Zhang, M. MicroTE: Fine grained traffic engineering for data centers. In Proceedings of the CONEXT, Tokyo, Japan, 6–9 December 2011.
23. Hindman, B.; Konwinski, A.; Zaharia, M.; Ghodsi, A.; Joseph, A.D.; Katz, R.; Shenker, S.; Stoica, I. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012; pp. 429–483.
24. Curtis, A.R.; Kim, W.; Yalagandula, P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In Proceedings of the 2011 Proceedings IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 1629–1637.
25. Kanagavelu, R.; Mingjie, L.N.; Mi, K.M.; Lee, B.; Francis; Heryandi. OpenFlow based control for re-routing with differentiated flows in Data Center Networks. In Proceedings of the 18th IEEE International Conference on Networks, Singapore, 12–14 December 2012; pp. 228–233.
26. Khurshid, A.; Zou, X.; Zhou, W.; Caesar, M.; Godfrey, P.B. Veriflow: Verifying network-wide invariants in real time. *ACM SIGCOMM Comput. Commun. Rev.* **2012**, *42*, 467–472. [CrossRef]
27. Tso, F.P.; Pezaros, D.P. Baatdaat: Measurement-based flow scheduling for cloud data centers. In Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC), Split, Croatia, 7–10 July 2013.
28. Li, J.; Chang, X.; Ren, Y.; Zhang, Z.; Wang, G. An Effective Path Load Balancing Mechanism Based on SDN. In Proceedings of the IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, Beijing, China, 24–26 September 2014; pp. 527–533.
29. Li, D.; Wang, S.; Zhu, K.; Xia, S. A survey of network update in SDN. *Front. Comput. Sci.* **2017**, *11*, 4–12. [CrossRef]
30. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 3–14. [CrossRef]
31. Alizadeh, M.; Atikoglu, B.; Kabbani, A.; Lakshmikantha, A.; Pan, R.; Prabhakar, B.; Seaman, M. Data center transport mechanisms: Congestion control theory and IEEE standardization. In Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing, Urbana-Champaign, IL, USA, 23–26 September 2008; pp. 1270–1277.
32. Duan, Q.; Ansari, N.; Toy, M. Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Netw.* **2016**, *30*, 10–16. [CrossRef]
33. Zhong, H.; Fang, Y.; Cui, J. Reprint of "LBBSRT: An efficient SDN load balancing scheme based on server response time". *Futur. Gener. Comput. Syst.* **2018**, *80*, 409–416. [CrossRef]
34. Shu, R.; Ren, F.; Zhang, J.; Zhang, T.; Lin, C. Analysing and improving convergence of quantized congestion notification in Data Center Ethernet. *Comput. Netw.* **2018**, *130*, 51–64. [CrossRef]