Hindawi
Scientific Programming
Volume 2017, Article ID 3579540, 8 pages
https://doi.org/10.1155/2017/3579540

Research Article

Software-Defined Congestion Control Algorithm for IP Networks

Yao Hu, Ting Peng, and Lianming Zhang

College of Physics and Information Science, Key Laboratory of Internet of Things Technology and Application, Hunan Normal University, Changsha 410081, China

Correspondence should be addressed to Lianming Zhang; zlm@hunnu.edu.cn

Received 7 October 2017; Revised 28 November 2017; Accepted 11 December 2017; Published 27 December 2017

Academic Editor: Basilio B. Fraguela

The rapid evolution of computer networks, increase in the number of Internet users, and popularity of multimedia applications have exacerbated the congestion control problem. Congestion control is a key factor in ensuring network stability and robustness. When the underlying network and flow information are unknown, the transmission control protocol (TCP) must increase or reduce the size of the congestion window to adjust to the changes of traffic in the Internet Protocol (IP) network. However, it is possible that a software-defined approach can relieve the network congestion problem more efficiently. This approach has the characteristic of centralized control and can obtain a global topology for unified network management. In this paper, we propose a software-defined congestion control (SDCC) algorithm for an IP network. We consider the difference between TCP and the user datagram protocol (UDP) and propose a new method to judge node congestion. We initially apply the congestion control mechanism in the congested nodes and then optimize the link utilization to control network congestion.

1. Introduction

In recent years, the development of science and technology has increased user demand for computer networks, which has accelerated the severity of network congestion problems. The increase in devices having applications that require file downloading and sharing, Internet browsing, voice over Internet phone, and multimedia has also contributed to the severity of network congestion problems. Thus, efficient congestion control is a key factor to ensure IP network stability and robustness.

There are two approaches in solving the network congestion problem. The first is the end-to-end approach, where research is focused on the mechanism of terminal congestion control to relieve congestion. The second is the network side approach, where congestion control is achieved through data stream scheduling and line management on the forwarding nodes. The initial research focused on the end-to-end congestion control, such as the widespread use of transmission control protocol (TCP). Because of its good adaptability and extensible capability, TCP has received widespread interest and became the main congestion control algorithm presently. Researchers in this area have conducted significant exploration and investigation and have put forward many improvements. For example, the latest versions of the four kinds of TCP congestion control algorithms, namely, Veno, Westwood, Hybla, and Cubic, have achieved optimization by packet loss judgment, bandwidth prediction, time delay compensation, and homogeneous compensation mechanism, respectively [1–3]. Paper [4] has studied the micro-level behaviour characteristics of TCP congestion control algorithm in multihop wireless networks. Paper [5] has proposed an adaptive cross-layer-based TCP congestion control for 4G wireless mobile cloud access. A deadline-aware congestion control mechanism, based on a parametrization of the traditional TCP New Reno congestion control strategy, has been proposed in [6].

The appropriate limit of the biggest transfer rate of each data flow can effectively reduce network congestion and packet loss. However, previous studies reported that when the underlying network and information flow are unknown, TCP must increase or reduce the size of congestion window to adjust to the changes of traffic [7–9]. This finding highlighted that the traditional IP network lacks the direct control of forwarding queue and cannot guarantee link utilization and quality of service. We have considered this aspect in our research to find a solution for the network congestion problem.

In the traditional IP network architecture, the control and forwarding planes are highly integrated, and the network is difficult to extend and customize. It has a long technology update cycle and is too dependent on network equipment manufacturers. Simultaneously, the increasingly complex network environment makes it difficult to develop the network and make innovations related to the hardware of physical devices or software of related protocols and performances, especially in the proprietary equipment and closed interfaces. Programmable models and code that allow data transmission from the network to the application are needed. The technology of software-defined everything from applications to infrastructure is one of the top 10 strategic technology trends identified by Gartner [10]. Software-defined networking (SDN) separates the network control and forwarding functions, enabling the network to be directly programmable, dynamic, and manageable. The separation of the control plane and data plane allows the core technology OpenFlow to realize the flexible control of the network traffic, making the network more intelligent, and provide a better platform for network application innovation [11–13].

In this study, we used a software-defined approach to solve the congestion control problem in the IP network. Compared with the existing literature, this paper has the following major contributions:

(i) A comprehensive survey of congestion control in the IP network is presented. In the traditional networks, the underlying network and flow information are unknown. It must increase or reduce the size of the congestion window to adjust to the changes of traffic, while a software-defined approach can relieve the network congestion problem more efficiently. So, we propose a software-defined congestion control (SDCC) algorithm for the IP network.

(ii) The type of data flow in the IP network is taken into account. The difference between TCP and UDP is pointed out. Based on the different characteristics of TCP and UDP, we propose a new method in judging congested nodes.

(iii) The simulation of our algorithm is comprehensively explained. We consider the packet loss rate and optimization of link utilization. The simulation shows that the proposed algorithm achieves network congestion control and optimizes the link utilization.

In the following sections, we first discuss the existing research on the congestion problem in the IP network based on SDN in Section 2. Section 3 presents an SDCC algorithm for the IP network. The effectiveness of the algorithm is verified by a simulation experiment, described in Section 4. Finally, the paper concludes in Section 5.

2. Background

2.1. Related Work. Network congestion control using a software-defined approach is still in its early stage of development. Ghobadi et al. [14] proposed a TCP dynamic adjustment framework based on SDN-OpenTCP. OpenTCP is an SDN application deployed on an SDN controller using the global network view of SDNs and is established based on the good strategy mechanism with the forwarding nodes and modifying the terminal protocol stack to continue the dynamic adjustment with the terminal congestion mechanism to achieve optimization. It fills the SDN application gap in the field of congestion control and is very interesting. However, OpenTCP is a new architecture that is widely deployed and needs to modify its source side, forwarding nodes, controllers, and other global network to provide support. This framework also illustrates that the TCP dynamic management strategy is open to managers and does not have a specific adjustment method and adjustment of the target. Long et al. [15] proposed a kind of network congestion control mechanism based on OpenFlow, which is considered as the first SDN standard. This mechanism is based on the OpenFlow network features that define a method in judging the link congestion control threshold. When network congestion occurs, the corresponding congestion control mechanism is immediately started, and the maximum bandwidth data flow in the congested link is identified and rerouted to achieve load balance. Lu et al. [16] presented a congestion control algorithm that supports multiple active queue management. The algorithm is dynamically adaptive to interactive link information and can effectively use bandwidth and network congestion control. Gholami and Akbari [17] proposed a method based on SDN and OpenFlow to solve this problem. In this method, link congestion is detected by central monitoring of the port statistics of the OpenFlow enabled switches, and some of the flows in any congested link are rerouted by the OpenFlow controller. This method highlighted the effectiveness of SDN in solving the congestion problem. Hershberger et al. [18] also proposed a new congestion optimization algorithm based on SDN. This algorithm chooses the k shortest path for each data flow ahead of schedule. When network congestion occurs, the k paths are sequentially traversed to determine which path is not congested. This method realizes the load balance to some extent; however, it also has a significant computation overhead. Meanwhile, these methods do not consider the type of data flow. If the selected routing involves a UDP data flow with highly demanding timeline but allows packet loss, and the delay of the new reroute shortest path is large. Then, this reroute strategy may not be an optimal choice. Therefore, we considered the difference of the type of data flow in our proposed algorithm.

2.2. Key Principles of a Software-Defined Approach. SDN is a mature software-defined area that originated from Stanford University's Clean Slate research [19, 20]. It is considered a new kind of network architecture and a method of implementing network virtualization. Its core concept is to separate the control plane and the data plane of the network hardware equipment. Software programming is implemented in the control plane. The SDN architecture can be divided into three levels: (1) application layer, (2) control layer, and (3) equipment layer. It has the following three basic characteristics: (1) centralized control, (2) programmable software, and (3) network virtualization.

OpenFlow network is comprised of the OpenFlow switch and OpenFlow controller. Switches and controllers establish the secure connection by transport layer security (TLS) or TCP prior to OpenFlow message interaction, issuing flow sheets, information query, reporting interface status, and other functions.

In OpenFlow network, controllers first send the link layer discovery protocol (LLDP) message (encapsulated in the Packet-Out message sent) to each of the OpenFlow switches to obtain the global network topology. Then, the switches regularly send Port-Status messages to controllers to report port information of each switch. This information, including data flow header and transmission rate, is recorded in the data information flow of each port in the switches.

3. Software-Defined Congestion Control Algorithm

This paper investigates the use of software-defined approach in solving the congestion control problem in the IP network. We proposed an SDCC algorithm that considered the difference between TCP and UDP, a new method in judging congested nodes, and the start of the congestion control mechanism in the congested nodes, then controlling the network congestion to optimize the link utilization.

The main concept of the SDCC algorithm is based on the SDN architecture, where the controller obtains port information of each switch to monitor for congested links. If network congestion exists, the congestion control mechanism is started, then one or more appropriate data flows are selected, and the shortest rerouting path is calculated to control the congestion. This process allows the efficient use of network bandwidth resources and increases the link utilization.

We considered the data flow condition in the real network and mainly focused on the heterogeneity between the TCP data flow and the UDP data flow in proposing this congestion control mechanism.

The process is as follows: (1) the controller obtains the global information and regularly monitors the network congestion condition; (2) estimate whether there are congested links based on formula (1); (3) if congested link is discovered, it is removed from the topology image to generate a new topology image; (4) calculate the shortest path between the last node of the congested node to the destination node; and (5) select one or more appropriate data flows to reroute.

3.1. Estimating Congested Links. In the OpenFlow network, the controller, according to the OpenFlow v1.0 standard specification, obtains a global network topology image through the LLDP and then sends port status information request message to each of the OpenFlow switches regularly. OpenFlow switches receive this message and feedback to the controller the data information flow of each port in the switches, such as data flow header and transmission rate.

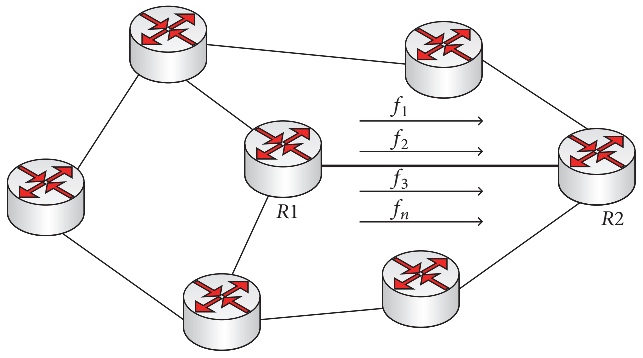

Figure 1: Bottom switch network.

```
Input: link bandwidth
Output: congested links
(1)   For $f_k$ in link
(2)       IF flow = TCP
(3)           $B_{tcp} = B_{tcp} + flow$
(4)       ELSE $B_{udp} = B_{udp} + flow$
(5)   IF $B_{th}$-$B_{udp} < B_{tcp} < B$
(6)       link → congested
(7)   ELSE continue monitoring
```

Algorithm 1: Estimating.

As shown in Figure 1, the controller detects that, in the OpenFlow network, there are n data flows $(f_1, f_2, f_3, \ldots, f_n)$ in R1 to R2 link, and the rate of each data stream is $(v_1, v_2, v_3, \ldots, v_n)$. Moreover, there are i UDP data flows and j TCP data flows.

Assume that the total bandwidth for the link R1 to R2 is B, and the link congestion threshold is B_{th} (specific value of B_{th} is according to the physical properties of the network environment). B_{tcp} is the congestion threshold of the TCP flows in this link, which is the link surplus maximum bandwidth for the TCP data flows after meeting the UDP data flow transmission demand as follows:

$$\begin{gathered} B'_{th}(t) = B_{th} - \sum_{k=1}^{i} f_k^{udp}(t) \\ B_{tcp} = \sum_{k=1}^{j} f_k^{tcp}(t), \end{gathered} \tag{1}$$

where $i + j = n$, B_{tcp} is the total rate of the TCP flows, and the initial value of B_{tcp} and B_{udp} is 0. Then, if $B'_{th} < B_{tcp} < B$, we confirm that R2 is in a congestion state, and the link R1 to R2 is the congested link. The algorithm for estimating congested links is as in Algorithm 1.

3.2. Calculation of Rerouting Links. Now, the controller needs to choose a new path for rerouting part of the data flows in the congested link. The controller removes all estimated

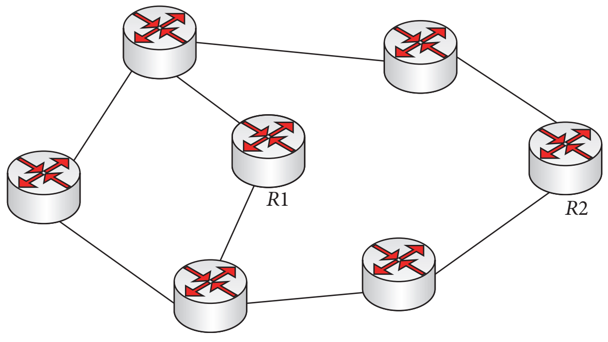

Figure 2: New bottom switch network.

```
Input: link and node distance
Output: new path and update the flow table
(1)   IF link[$a$]→ congested
(2)       dist[$a$] = INFINITY
(3)   For $i$ do
(4)       dist[$i$] = $v[0] \rightarrow v[i]$
(5)   For $i$ do //find the nearest point
(6)       int mindist = INFINITY
(7)       For $j$ do
(8)           If dist[$j$] < mindist
(9)               mindist = dist[$k$]
(10)      For $j$ do //update the shortest path
(11)          If mindist + $v[k] \rightarrow v[j] < v[0] \rightarrow v[j]$
(12)          dist[$j$] = mindist + $v[k] \rightarrow v[j]$
(13)  Newlink → Packet_out //update flowtable
(14)  OFPFC_DELETE
(15)  OFPFC_ADD
(16)  Delete OutFlowtable
```

Algorithm 2: Rerouting.

congested links from the global network topology image and then obtains a new noncongested network topology image as shown in Figure 2. In the concrete execution, we set the link path distance as infinity to achieve the deletion.

Then, using the shortest path algorithm to calculate the shortest path with non-congestion from the last node of the congested node to the destination node, and finally choosing the appropriate data flows transferred to this new path to complete the transmission, the algorithm of the OpenFlow network rerouting is presented as in Algorithm 2. Parameter i represents i UDP data flows, and parameter j represents j TCP data flows. Parameter k represents the k_{th} data flow.

3.3. Selection of Rerouted Data Flows. Considering that the UDP data flows always emphasize short transfer time and high-efficiency requirements, we selected the TCP data flows for the rerouting. Assuming that there are j TCP data flows $(f_1, f_2, f_3, \ldots, f_j)$ on one congested link, the rate of each data flows is $(v_1, v_2, v_3, \ldots, v_j)$, and we obtain the following:

$$B'_{\text{tcp}} = B_{\text{tcp}} - (v_a + v_b + \cdots + v_k). \tag{2}$$

To ensure high link bandwidth utilization, when the total rate of the TCP flows on this link B_{tcp} minus part of the rate $(f_a, f_b, \ldots, f_k)$, and a new total rate of TCP flows B'_{tcp} closest to B'_{th} is obtained, then we are assured that the data flows $(f_a, f_b, \ldots, f_k)$ are optimal choice for rerouting the transmission. The selection algorithm of the rerouted data flows are as in Algorithm 3. Parameter w represents the number of the most suitable data flows.

4. Experiment and Result

4.1. Experiment Design. We considered the data flow condition in the real network and mainly focused on the heterogeneity between the TCP data flow and the UDP data flow in proposing this congestion control mechanism. In our test environment, the controller obtains the global information and regularly monitors the network congestion condition. The controller estimates whether there are congested links. If congested link is discovered, it is removed from the topology image to generate a new topology image. The controller calculates the shortest path between the last node of the congested node to the destination node. At last, the controller selects one or more appropriate data flows to reroute.

To verify the proposed congestion control mechanism using the OpenFlow network, we built the test environment as shown in Figure 3. OFS1, OFS2, and OFS3 are OpenFlow switches by Mininet simulation; OFC is the OpenFlow controller with POX [21]; PC1–PC6 are common user terminals, and we use the Iperf tool on user terminals to send the data flows.

We added the link monitoring, congestion detection, rerouted data flow selection, and shortest path calculation function modules in the POX controller and combined them with the existing modules of topology discovery and flow table update to achieve the proposed congestion control mechanism. The running process of each module is shown in Figure 4.

After the POX controller starts, the LLDP protocol is first run to obtain the entire network topology, and then the congestion detection module is triggered to regularly (the experiment is set at 2 s) query each OFS for status information of each port and record the number and size of each flow through this port. Then, the controller will add the rates of all data flows together. We found that the NetFPGA's port line speed is 1 Gbps. After many experiments, when the port rate reaches 950 MB/s, the packet loss rate will quickly reach 2% magnitude, and it is out of the standard of common business demand. Therefore, to simulate the real network environment, we set the port congestion threshold at 950 MB/s in our test.

4.2. Results and Analysis. To facilitate testing, we built four data flows with Iperf (as shown in Table 1), and sets OFS1-OFS2 as Path 1 and OFS1-OFS3-OFS2 as Path 2. As identified by the shortest path algorithm, these four flows will be transmitted through Path 1. In this experiment, the rate of $f3$ will be increased over time.

```
Input: all data flow rates through the congested link
Output: number of selected flow
(1)   For i do
(2)       sum = sum + flow[i]
(3)   For i do //Select the appropriate flow and tag it
(4)       If flow = TCP && congested && sum-flow[w] > bestsum
(5)           bestsum = sum − flowsize[w]
(6)           bestflownum = w
(7)   Return bestflownum
```

Algorithm 3: Selection.

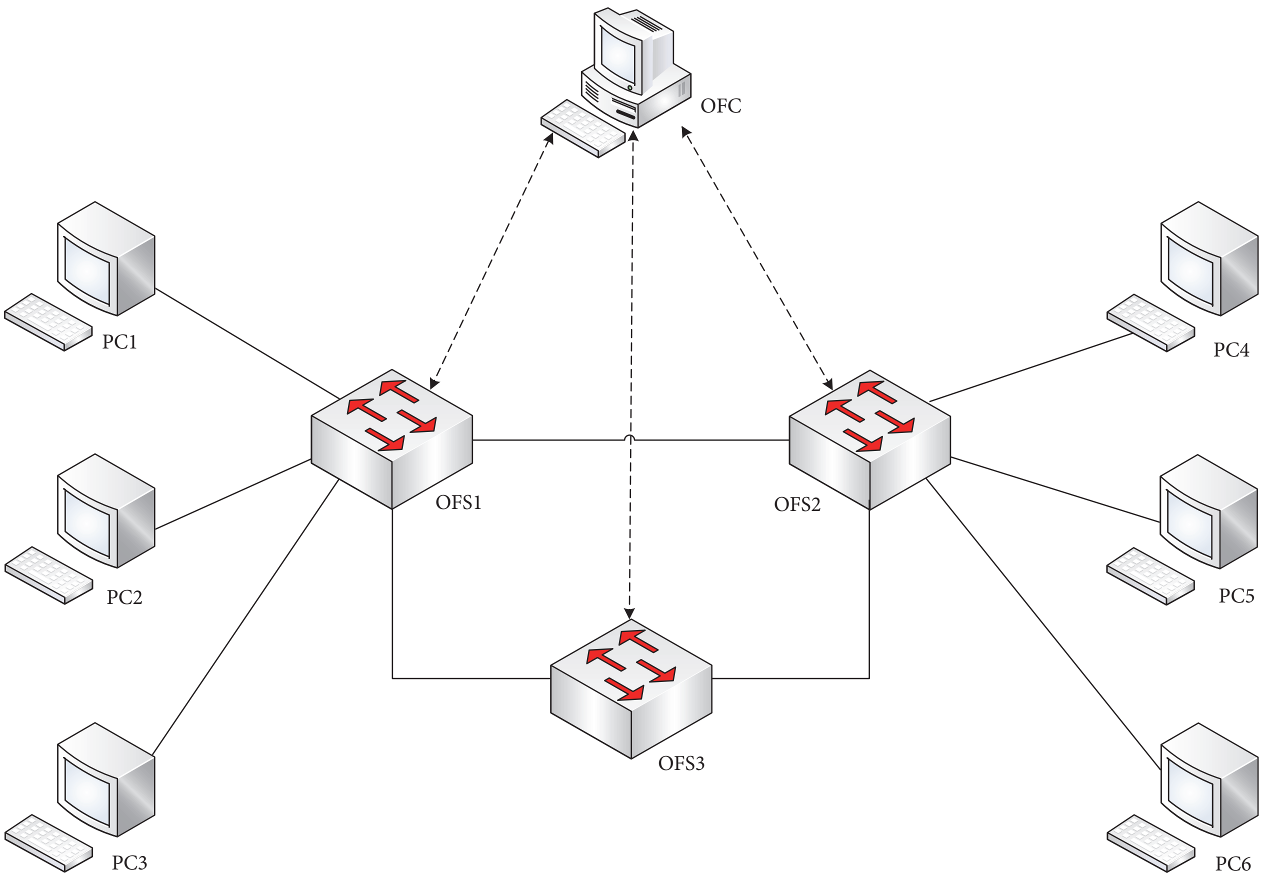

Figure 3: OpenFlow network experiment platform.

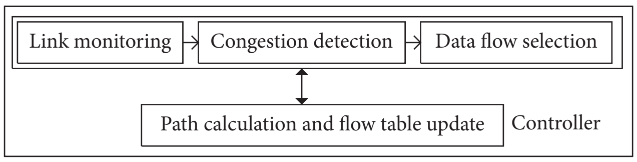

Figure 4: Module architecture of the POX controller in SDCC algorithm.

Table 1: Set of four data flows.

Flow	Type	Rate
$f1$	TCP (PC1–PC4)	150 MB/s
$f2$	UDP (PC2–PC4)	100 MB/s
$f3$	TCP (PC3–PC6)	Increasing
$f4$	UDP (PC1–PC6)	350 MB/s

(1) Link Bandwidth. Figure 5 shows the complete process of Path 1 and Path 2 loading with $f3$ data flow rate. First, the two links maintain a stable trend without large fluctuation. When the $f3$ growth rate reaches around 350 MB/s, an obvious turning point occurred as shown in Figure 6. At this time, the OFC detected OFS2 link is congested, and the most appropriate data flow is selected ($f1$ in the experiment) from all data flows through this link. Finally, the flow is rerouted to

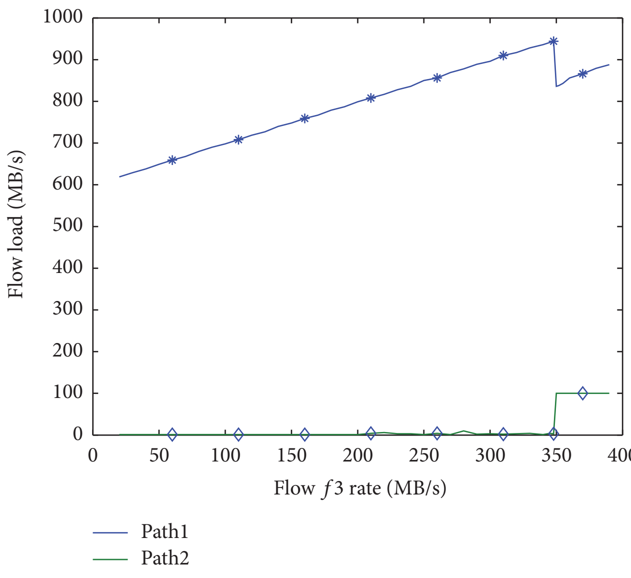

Figure 5: Link bandwidth.

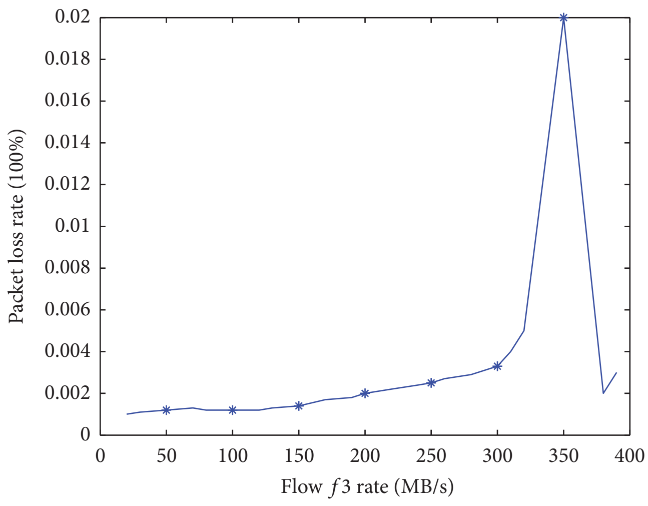

Figure 7: Packet loss rate.

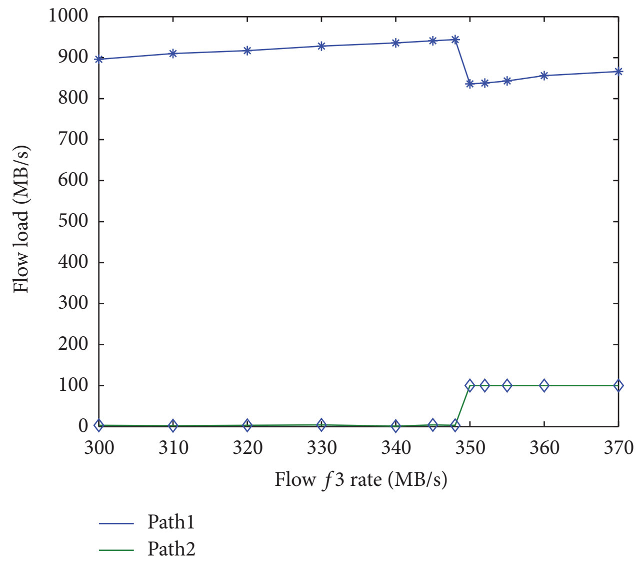

Figure 6: Link bandwidth in turning point.

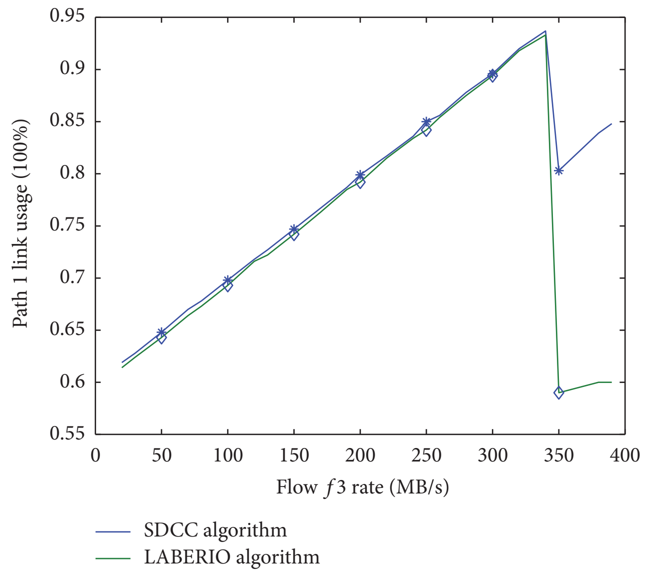

Figure 8: Link utilization.

Path 2 because we only select TCP data flows to reroute in our algorithm, and the UDP data flows are not adjusted consistent with the aim of the algorithm.

Figure 6 provides details of the turning point of the link bandwidth. Then, Path 1 loading decreases by 150 MB/s from 950 MB/s to 800 MB/s. Moreover, Path 2 significantly increased in loading size at about 150 MB/s, which explains the transmission of $f1$ to Path 2. After the congestion relief, the loading in Path 1 continues to rise steadily with the increase of $f3$ data flow until the next congestion.

(2) Packet Loss Rate. The packet loss rate is shown in Figure 7, where the increase of $f3$ increases the packet loss rate of Path 1, but the overall trend and value are small and within a reasonable range. When $f3$ reached 350 MB/s, an obvious increase in packet loss rate was observed. The packet loss rate has increased dramatically to 2%, which indicates that it has reached a state of congestion at this time. While the rate of $f3$ increases from 350 MB/s to 400 MB/s, there is an obvious decline process, which shows the congestion relief and the packet loss rate return to a reasonable range.

(3) Link Utilization. At the full stage of flow $f3$ rate from 0 to 400 MB/s, the average link utilization of the proposed algorithm is about 78.5%, and the LABERIO algorithm has an average link utilization of about 74%. Figure 8 shows that when congestion did not occur, the link utilization rate does not have much difference. However, when congestion occurs and congestion control is performed, the SDCC algorithm proposed in this paper does not fluctuate significantly. Moreover, the algorithm still maintained high link utilization. First,

the utilization on Path 1 link starts at around 60% and shows a linear growth trend with the flow rate of flow $f3$, where the maximum link utilization can be up to 95%. When the rate of flow $f3$ reaches 350 MB/s, the link utilization of the SDCC algorithm is about 80%. Moreover, a follow-up with the growth rate of flow $f3$ showed an upward trend. However, the LABERIO algorithm significantly declined at less than 60%, and the follow-up showed almost no growth trend. Therefore, the SDCC algorithm is superior to the LABERIO algorithm in link utilization.

(4) Complexity. Finally, the complexity analysis shows the considerable properties of the SDCC algorithm that are mainly related to the calculation of three parts, namely, the real-time monitoring and judgment of the congested link, the new shortest path calculation of rerouting, and the selection of rerouting data flows. If k nodes exist in a network topology and there are n data streams on the link, the time complexity of these three parts is $O(nk)$, $O(k^2)$, and $O(n)$.

Therefore, the results show the effectiveness of our congestion mechanism. It effectively relieved congestion and guaranteed the packet loss rate and link utilization.

5. Conclusion and Future Work

Congestion is a common problem in the IP network, but the current level of development in the traditional network algorithm is insufficient to meet the growing network demand. A software-defined approach is required to find a solution to the network congestion problem. Using centralized control features of the SDN network, we proposed a new network congestion control algorithm called SDCC algorithm. The SDCC algorithm is based on the SDN architecture, and it can obtain a global topology for unified network management. SDCC algorithm can relieve the network congestion problem more efficiently. We considered the difference between TCP and UDP and proposed a new method in judging congested nodes. Moreover, the simulation shows the effectiveness of our congestion mechanism. The proposed algorithm achieves network congestion control, guarantees the packet loss rate, and optimizes the link utilization. In future work, we will apply a software-defined approach to the data center network or other more complex and special network to solve the congestion problem in depth. Furthermore, we can still optimize our algorithm considering the different attributes of UDP packets or other such factors and adding different parameters to provide a fair and accurate data flow selection strategy.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by grants from the National Natural Science Foundation of China (61572191 and 61402170).

References

[1] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, 2003.

[2] C. Caini and R. Firrincieli, "TCP Hybla: A TCP enhancement for heterogeneous networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, 2004.

[3] C. E. Palazzi, M. Brunati, and M. Roccetti, "An OpenWRT solution for future wireless homes," in *Proceedings of the 2010 IEEE International Conference on Multimedia and Expo, ICME 2010*, pp. 1701–1706, July 2010.

[4] D. SreeArthi, S. Malini, M. J. Jude, and V. C. Diniesh, "Micro level analysis of TCP congestion control algorithm in multi-hop wireless networks," in *Proceedings of the 2017 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5, January 2017.

[5] B.-J. Chang, Y.-H. Liang, and J.-Y. Jin, "Adaptive cross-layer-based TCP congestion control for 4G wireless mobile cloud access," in *Proceedings of the 3rd IEEE International Conference on Consumer Electronics-Taiwan, ICCE-TW 2016*, pp. 1-2, May 2016.

[6] M. Claeys, N. Bouten, D. De Vleeschauwer et al., "Deadline-aware TCP congestion control for video streaming services," in *Proceedings of the 12th International Conference on Network and Service Management*, pp. 100–108, November 2016.

[7] J. Gruen, M. Karl, and T. Herfet, "Network supported congestion avoidance in software-defined networks," in *Proceedings of the 2013 19th IEEE International Conference on Networks, ICON 2013*, p. 16, December 2013.

[8] A. El-Mougy, M. Ibnkahla, G. Hattab, and W. Ejaz, "Reconfigurable wireless networks," *Proceedings of the IEEE*, vol. 103, no. 7, pp. 1125–1158, 2015.

[9] J.-H. Wang, K. Ren, W.-G. Sun, L. Zhao, H.-S. Zhong, and K. Xu, "Effects of iodinated contrast agents on renal oxygenation level determined by blood oxygenation level dependent magnetic resonance imaging in rabbit models of type 1 and type 2 diabetic nephropathy," *BMC Nephrology*, vol. 15, no. 1, article 140, 2014.

[10] "Gartner Identifies the Top 10 Strategic Technology Trends for 2015," 2014, http://www.gartner.com/newsroom/id/2867917.

[11] P. Sun, M. Yu, M. J. Freedman, J. Rexford, and D. Walker, "HONE: Joint Host-Network Traffic Management in Software-Defined Networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 374–399, 2015.

[12] Z. Ge, R. Gu, and Y. Ji, "An active queue management adaptation framework for software defined optical network," in *Proceedings of the 2014 13th International Conference on Optical Communications and Networks, ICOCN 2014*, p. 14, November 2014.

[13] L. Zhang, Q. Deng, Y. Su, and Y. Hu, "A Box-Covering-Based Routing Algorithm for Large-Scale SDNs," *IEEE Access*, vol. 5, pp. 4048–4056, 2017.

[14] M. Ghobadi, S. H. Yeganeh, and Y. Ganjali, "Rethinking end-to-end congestion control in software-defined networks," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks, HotNets 2012*, pp. 61–66, October 2012.

[15] H. Long, Y. Shen, M. Guo, and F. Tang, "LABERIO: dynamic load-balanced routing in OpenFlow-enabled networks," in *Proceedings of the IEEE International Conference on Advanced Information Networking Applications*, pp. 290–297, 2013.

[16] L. Lu, Y. Xiao, and H. Du, “OpenFlow control for cooperating AQM scheme,” in *Proceedings of the 2010 IEEE 10th International Conference on Signal Processing, ICSP2010*, pp. 2560–2563, October 2010.

[17] M. Gholami and B. Akbari, “Congestion control in software defined data center networks through flow rerouting,” in *Proceedings of the 23rd Iranian Conference on Electrical Engineering, ICEE 2015*, pp. 654–657, May 2015.

[18] J. Hershberger, M. Maxel, and S. Suri, “Finding the k shortest simple paths: A new algorithm and its implementation,” *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, Article ID 1290682, 2007.

[19] N. McKeown, T. Anderson, H. Balakrishnan et al., “OpenFlow: enabling innovation in campus networks,” *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[20] M.-K. Shin, K.-H. Nam, and H.-J. Kim, “Software-defined networking (SDN): A reference architecture and open APIs,” in *Proceedings of the 2012 International Conference on ICT Convergence: “Global Open Innovation Summit for Smart ICT Convergence”, ICTC 2012*, pp. 360-361, October 2012.

[21] POX, 2017, https://github.com/noxrepo/.