Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 1819202, 7 pages
https://doi.org/10.1155/2022/1819202

Research Article

A Traffic Scheduling Method Based on SDN

Li Wang

China Oil & Gas Pipeline Network Corporation, Beijing 312000, China

Correspondence should be addressed to Li Wang; b20160502106@stu.ccsu.edu.cn

Received 23 February 2022; Revised 21 March 2022; Accepted 1 April 2022; Published 30 April 2022

Academic Editor: Dinesh Kumar Saini

The rise of big data has brought severe challenges to data storage, which also means that computing resources and storage resources have become centralized. In the past, business data could end up after being processed locally or at the secondary node. Now, all data must be uploaded to each data center for unified processing. Data collection also requires users to access data center services and then send them back, which undoubtedly increases the pressure on the backbone network. This also means that the annual private line cost of the enterprise will increase with the growth of network traffic. Large enterprises often use multiple lines between network nodes to improve link utilization and network reliability. However, in most cases, the line utilization of the backup line is usually low, because the traditional routing algorithm forwards according to the optimal path. Load sharing or traffic scheduling is achieved by adjusting interface cost values or BGP parameters. In large networks, it often affects the entire body. Each adjustment requires detailed planning and validation. At the same time, due to the diversity of applications, different applications have different business characteristics. If the network can distinguish different applications, sense network conditions, and adjust the QoS and traffic paths of applications as needed, the link utilization of leased lines and the network quality of applications will be significantly improved. SDN separates the control plane from the data plane and intelligently controls the network forwarding policy through the global controller, which improves the flexibility and security of the network. Based on the traditional large-scale enterprise WAN architecture, this paper proposes a complete traffic scheduling solution based on SDN to achieve secure, controllable, and flexible scheduling of enterprise WAN network resources.

1. Introduction

With the rapid development of cloud computing, big data, mobile applications, and the Internet of things, the traffic on the WAN will also increase explosively, and the expenditure of enterprises on dedicated line fees will also rise sharply. However, the enterprise's dedicated line links often have low average utilization but high peak utilization, and the enterprise's redundant links cannot be effectively utilized. According to the prediction of authoritative institutions, the future traffic will continue to grow at a faster rate than expected. In order to cope with such growth, it is very expensive for enterprises to upgrade the bandwidth of leased lines and build and use free optical cables. Enterprises also hope to effectively control the bandwidth cost caused by the increase in bandwidth demand.

With the rapid growth of network traffic, some problems in the enterprise network are gradually exposed: at present, many enterprise networks are often unable to achieve comprehensive visualization, making the network business slow and unable to locate quickly; enterprises spend a lot of money on the improvement of dedicated line bandwidth, but in many cases, the service quality is still not improved. Therefore, enterprises urgently need to solve the problems in the existing network through the transformation of the network and technological innovation.

In terms of network visualization, the traditional network monitoring scheme is often realized through SNMP [1], which has complex configuration, low sampling frequency, and few monitoring indicators. The collected monitoring data are often only an auxiliary means to troubleshoot and solve problems and cannot be used as the basis for network optimization. Therefore, enterprises need

to use some new monitoring technologies, such as bgp-ls to realize the automatic drawing of network topology [2], telemetry to realize the second push of equipment performance indicators, and network probe to realize the accurate analysis of traffic [3]. Through a large number of monitoring data and analyzing the performance indicators of each service, the intelligent scheduling of network traffic is realized.

In terms of network traffic scheduling, traditional network engineers often realize service traffic scheduling by manually configuring MPLS-TE [4]. Its configuration is complex and error-prone. With the increase of enterprise network scale, its configuration often leads the whole body, which has high requirements for network engineers. Google's B4 pioneered the use of SDN to realize the service traffic scheduling between global data centers, which greatly improved the bandwidth utilization of its submarine optical cable and reduced the annual cost of the link cost [5]. At present, large enterprises are also facing this situation. If enterprises can use SDN technology to centrally control network business traffic, ensure important business operation, and improve bandwidth utilization, it will greatly reduce the leased line rental cost of enterprises.

This paper is divided into four parts. Section 1 introduces the overall scheme of SDN traffic scheduling. Section 2 introduces the design and implementation of the scheduling algorithm. Section 3 tests and verifies the scheduling algorithm. Section 4 is the summary and prospect.

The overall solution is mainly divided into three parts, including network visualization, traffic scheduling, and configuration distribution. The purpose of network visualization is to obtain the data and indicators of the whole network and provide a basic data source for the calculation of traffic scheduling. Traffic scheduling refers to the algorithms used in Google B4 [6] network (KSP algorithm, greedy algorithm, maximum, and minimum fair value algorithm [7]). According to the collected network indicators, application priority, network topology, and other information, each application is allocated its bandwidth, path, and priority globally and fairly [6]. The configuration distribution module generates the corresponding configuration according to the final traffic scheduling result and distributes it to the network device.

Network visualization mainly includes two parts: network topology visualization and network traffic visualization. Network topology visualization solves the problem of constructing the adjacency relationship between network nodes and links, and network traffic visualization solves the problem of network traffic data collection, analysis, and visualization.

At present, the most mainstream solution for network topology visualization is to obtain network topology information through BGP-LS (border gateway protocol link state). By extending the two attributes of address family and routing of BGP protocol, it carries the information of topology nodes, link information, and routing information of each node in the network through three attributes: node, link, and routing prefix [8]. At the same time, it also supports cross-domain network topology generation. The controller only needs to establish a BGP-LS session with one device in each domain to draw the network topology information of the whole network.

There are traditional NetFlow schemes for network traffic visualization, port images combined with network probe schemes, and some technical schemes produced by various manufacturers to cooperate with their own controllers, such as IFIT. Because NetFlow appeared earlier, most network devices support this function. It can realize the monitoring based on the network quintuple. Because NetFlow relies on the device for network traffic analysis, when the traffic flow in the network is large, it has a high-performance loss to the device. Therefore, in many cases, it will reduce the loss of equipment performance by reducing its sampling rate, which leads to the inaccurate traffic data obtained by using NetFlow [9]. The solution provided by the manufacturer is like IFIT. Its essence is to sample and analyze network traffic data similar to NetFlow. Compared with NetFlow, it has the advantage that end-to-end traffic analysis can be realized through whole network deployment. The advantage of port mirroring combined with network probe is that it has little impact on the performance of network equipment and because it is full traffic collection and analysis, the collected traffic data are more accurate [10].

Traffic scheduling is mainly divided into two parts, including path calculation and path and bandwidth allocation. The purpose of path calculation is to create a traffic path in advance for the scheduling algorithm. Path and QoS allocation is to allocate the path and bandwidth for each application flow according to the maximum and minimum fair value algorithm according to the network path, link bandwidth, application priority, and required bandwidth, so as to achieve the global optimal effect.

The algorithm can calculate the K optimal paths without loops from a starting point to an end point. Generally speaking, the bandwidth of the enterprise's dedicated line link is much smaller than that used by the intranet line. Therefore, when using this algorithm, we only need to reduce the cost value of the high bandwidth link and increase the cost value of the low bandwidth link, plus the appropriate number of paths, we can generate a path covering all low bandwidth links.

Assuming that the number of nodes in the network topology is n and the node has m deviation points, in the kth iteration, we need $1/2\sum_{i=1}^{m}(N-i)^2 = 1/6qN^3$ insert operation and $\sum_{k=1}^{m}(N-k)^2 = 1/3qN^3$ comparison operation, where $0 < q \leq 1\circ$. Hence, the insertion operation to be performed by the whole algorithm is $1/6qKN^3$, the comparison operation is $1/3qKN^3$, and the additional space required is $N^2 + KN$.

Application path and bandwidth allocation: this paper uses the minimum and maximum fair value algorithm [11] to calculate the application path and bandwidth allocation. The implementation of specific scheduling and fair allocation algorithm is introduced in detail in Section 2.

The configuration distribution module mainly completes two things, including determining the content of configuration distribution and determining the protocol of configuration distribution. The content of configuration

Table 1: Time required for CLI script and NETCONF to complete a certain number of configurations.

	CLI (s)	NETCONF (s)
100	5.389	0.636
200	11.326	0.738
500	27.654	1.356
1000	56.276	2.153

distribution needs to be determined according to the actual situation of the current network. In networks that only support MPLS, path switching can be realized by MPLS-TE combined with PBR or CBTs. In networks supporting segment routing [12], application flow scheduling can be realized through SR-TE, SR policy combined with CBTs or PBTS. At present, only NETCONF [13] and CLI can be used to configure distribution. Compared with CLI, NETCONF has great advantages in distribution efficiency, configuration consistency, and development efficiency, but not all devices can support NETCONF protocol, and its device versatility is poor. Table 1 compares the CLI and NETCONF distribution efficiency. If the enterprise's ingress node cannot support the NETCONF protocol, the upper SDN controller needs to design a perfect distribution process when developing relevant modules to ensure the success of distribution and the rollback after distribution failure. Time required for CLI script and NETCONF to complete a certain number of configurations is shown in Table 1.

2. Design and Implementation of the Scheduling Algorithm

With the expansion of enterprise scale, there are many kinds of applications in the network, and the application characteristics of each application are also very different. Some applications are sensitive to delay and packet loss but do not require high bandwidth; some applications are sensitive to delay and packet loss but require high bandwidth; some applications have high requirements for delay, packet loss, and bandwidth. Therefore, if each application is divided into a policy, the number is very large, and it is difficult for network devices to schedule each application. In IP packet designing, the first three bits of the TOS field are used as IP priority. The larger the number from 0 to 7, the higher the importance of the service. Later, in order to identify more service types, a new definition of TOS was made in RFC 2474 [14], taking the first six bits as DSCP and reserving the last two bits, so that 0 ~ 63 service types can be represented. Since the priority field in the MPLS tag only contains 3 bits, it is consistent with the TOS field, and generally, 6 and 7 in the TOS are reserved for basic protocols such as routing protocols. Therefore, this paper implements scheduling according to six types of applications from 0 to 5.

Path calculation: in this paper, Yen algorithm [11] is used to calculate the path. The detailed algorithm flow is shown in Figure 1.

Time required for CLI script and NETCONF to complete a certain number of configurations is shown in Table 1.

Different enterprise nodes access various applications to form hundreds of application flow groups (FGs). Each FG contains source address, destination address, priority, path, bandwidth, and other QoS information. At the same time, there are multiple paths from one site to another. These paths are divided into a tunnel group (TG). Different application classes have different priority and bandwidth requirements. The fair value fs (fair share) is obtained by priority according to the required bandwidth. The bandwidth function of each application can be obtained by taking the fair value s as the horizontal axis and the bandwidth B as the vertical axis $B(S) = \begin{cases} W \times S, 0 \le S \le B_{\max}/W \\ B_{\max}, S > B_{\max}/W \end{cases}$, as shown in Figure 2(a) The slope shown by each line is the weight W applied. For an application stream group FG, which contains several application streams, its bandwidth function is $B_{\text{sum}}(S) = \sum_{k=1}^{n} B_k(S)$; we can use the linear superposition of each application stream bandwidth function to represent B. K(s) represents the bandwidth function of the kth application stream. As shown in Figure 2(b), the bandwidth function diagram of the final FG is composed of a group of broken lines.

If the bandwidth and path allocated by each FG are calculated by linear programming, it takes a long time and has no good scalability. Therefore, this paper calculates the final result by greedy algorithm combined with dichotomy.

The fair algorithm process and pseudocode are as follows:

(1) Since the number of application streams covered at the branch node is small, bandwidth allocation for the bottle-neck link is given priority at the branch node. During allocation, all cases are enumerated by permutation and combination. The initial value of minfs is 0 and maxfs is the maximum fair value in the application flow group FG. Curfs represent the currently allocated fair value, which is initially set to maxfs. The pseudocode is as follows:

```
if (allocate_end (maxFS)==true) {
    curFS=maxFS;
}else{
    curFS=(maxFS+midFS)/2;
}
while (maxFS-curFS>=0.01) {
    if(allocate_end(curFS)==true){
        midFS=curFS;
        curFS=(maxFS+minFS)/2;
        save();  //save the end path info
    }else{
    maxFS=curFS;
    curFS=(maxFS+minFS)/2;
    }
}
```

(2) The bandwidth of the bottleneck link at the core/convergence is allocated. The calculation results in the first step are sorted according to the allocated

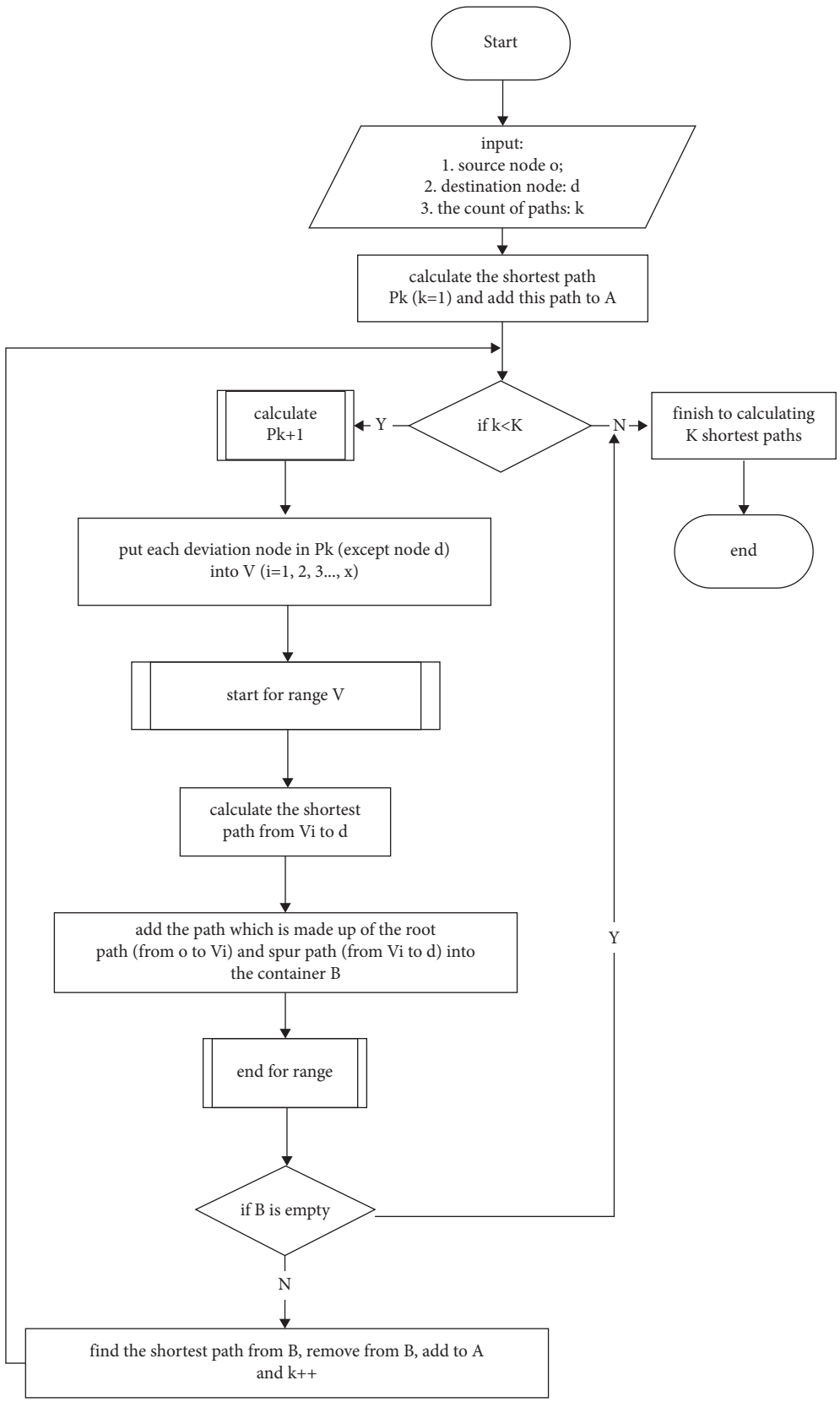

Figure 1: Implementation flow chart of Yen algorithm.

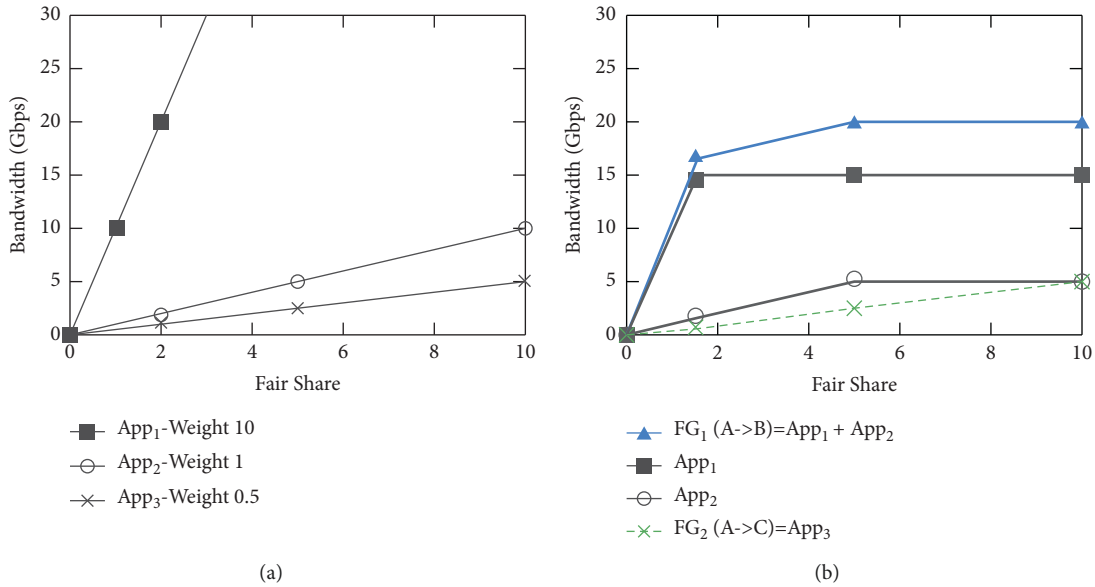

Figure 2: Bandwidth allocation function of the application stream group FG: (a) per-application and (b) FG-level composition.

bandwidth from large to small and then allocated to the links with large remaining bandwidth in all bottleneck lines. The exhaustive algorithm in the first step is not used here, mainly because the number of application streams at the core/aggregation link is relatively correct and the complexity of the exhaustive algorithm is high. Although the greedy algorithm used here is not the global optimal solution, its algorithm complexity is low and its error will decrease with the increase in the number of application streams. The maximum fair value maxfs is the maximum current fair value in all application streams (calculated in step 1), the minimum fair value minfs is 0, the current fair value curfs, flow_size is the number of all application streams, curbw [i] is the current bandwidth of the ith application stream, and priority [i] is the priority of application stream I. The pseudocode is as follows:

```
if (allocate_begin (maxFS)==true) {
    curFS=maxFS;
}else {
    curFS=(maxFS+midFS)/2;
}
while (maxFS-curFS>=0.01) {
    if(allocate_begin(curFS)==true){
        midFS=curFS;
        curFS=(maxFS+minFS)/2;
        save();  //save the start path info
    }else{
    maxFS=curFS;
    curFS=(maxFS+minFS)/2;
    }
}
//calculate the result
for(i=0;i<flow_size;i++){
    curBW[i]=min{curFS*priority[i],curBW[i]};
}
```

(3) The final path of the application flow is the path containing the bottleneck lines stored in steps 1 and 2.

3. Test and Result Analysis

This paper constructs a network topology, as shown in Figure 3: node *E* and node *f* are branch 1, nodes *g* and H are branch 2, and nodes I and *j* are branch 3. The link bandwidth is shown in the figure. Application flows are classified into five application classes. The path of flow1 is fixed as the optimal path, and flow5 is other nonimportant services. Its required bandwidth is the maximum bottleneck bandwidth value; that is, when meeting the bandwidth requirements of other important services, the remaining bandwidth can be allocated to this kind of applications.

The path from node a to each branch node is shown in Table 2. It can be seen that the path completely covers the bottleneck line.

By adjusting the required bandwidth in different scenarios, check whether the bandwidth is consistent with the expected bandwidth.

It can be seen from Table 3 that when the link bandwidth is greater than the required bandwidth of all application streams, when the link bandwidth is sufficient, the required bandwidth of each application stream will be met as much as possible.

Table 4 shows the results calculated by a fair algorithm when the link bandwidth is insufficient after adjusting the required bandwidth of high priority applications. It can be seen that the bandwidth requirements of high priority applications are guaranteed first, while the bandwidth of low priority applications can be allocated better when the link bandwidth is sufficient, and the bandwidth of bottleneck links can be fully used to improve the overall link utilization.

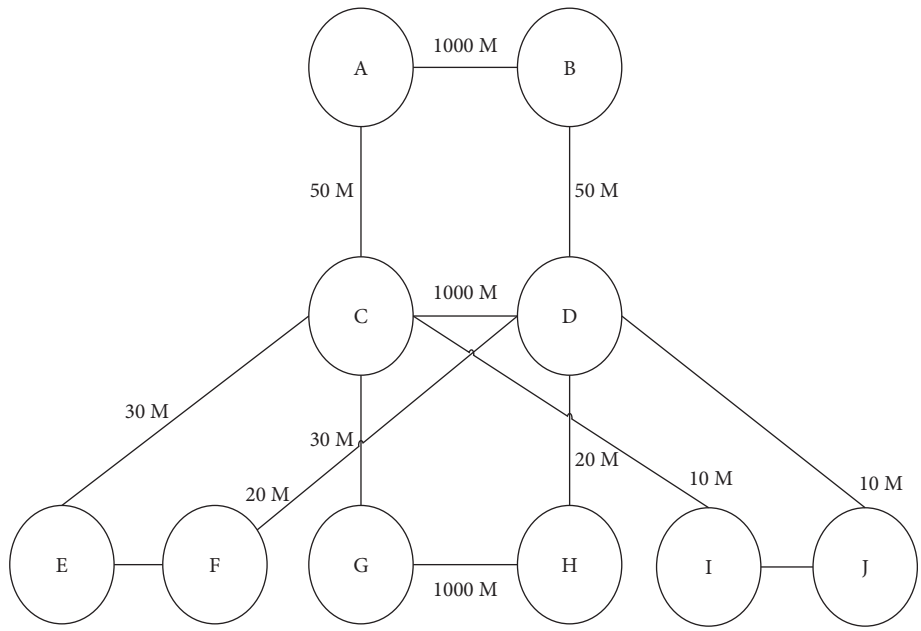

Figure 3: Example of network topology.

Table 2: All paths from node a to branch node.

Path name	Path detail
ETP1-path1	A- > c- > e
ETP1-path2	A- > c- > d- > f- > e
ETP1-path3	A- > b- > d- > c- > e
ETP1-path4	A- > b- > d- > f- > e
ETP2-path1	A- > c- > g
ETP2-path2	A- > c- > d- > h- > g
ETP2-path3	A- > b- > d- > c- > g
ETP2-path4	A- > b- > d- > h- > g
ETP3-path1	A- > c- > i
ETP3-path2	A- > c- > d- > j- > i
ETP3-path3	A- > b- > d- > c- > i
ETP3-path4	A- > b- > d- > j- > i

Table 3: Algorithm results when bandwidth demand is lower than link bandwidth.

Flow name	Calculated bandwidth (kbps)	Demand bandwidth (kbps)	Calculated path	Priority
ETP1_Flow1	4000	4000	ETP1-path1	6
ETP1_Flow2	8000	8000	ETP1-path4	5
ETP1_Flow3	8000	8000	ETP1-path3	4
ETP1_Flow4	2000	2000	ETP1-path3	3
ETP1_Flow5	20000	30000	ETP1-path2	1
ETP2_Flow1	4000	4000	ETP2-path1	6
ETP2_Flow2	4000	4000	ETP2-path4	5
ETP2_Flow3	8000	8000	ETP2-path4	4
ETP2_Flow4	2000	2000	ETP2-path3	3
ETP2_Flow5	20000	30000	ETP2-path1	1
ETP3_Flow1	4000	4000	ETP3-path1	6
ETP3_Flow2	4000	4000	ETP3-path1	5
ETP3_Flow3	2000	2000	ETP3-path1	4
ETP3_Flow4	2000	2000	ETP3-path4	3
ETP3_Flow5	8000	30000	ETP3-path4	1

Table 4: Algorithm calculation results after increasing the bandwidth required by high priority applications.

Flow name	Calculated bandwidth (kbps)	Demand bandwidth (kbps)	Calculated path	Priority
ETP1_Flow1	4000	4000	ETP1-path1	6
ETP1_Flow2	10000	10000	ETP1-path2	5
ETP1_Flow3	10000	10000	ETP1-path4	4
ETP1_Flow4	10000	10000	ETP1-path3	3
ETP1_Flow5	5750	30000	ETP1-path2	1
ETP2_Flow1	4000	4000	ETP2-path1	6
ETP2_Flow2	10000	10000	ETP2-path2	5
ETP2_Flow3	10000	10000	ETP2-path3	4
ETP2_Flow4	10000	10000	ETP2-path4	3
ETP2_Flow5	5750	30000	ETP2-path1	1
ETP3_Flow1	4000	4000	ETP3-path1	6
ETP3_Flow2	6000	10000	ETP3-path1	5
ETP3_Flow3	5000	10000	ETP3-path2	4
ETP3_Flow4	3750	10000	ETP3-path4	3
ETP3_Flow5	1250	30000	ETP3-path2	1

Table 5: The result of the etp2_ calculation of the boosting algorithm.

Flow name	Calculated bandwidth (kbps)	Demand bandwidth (kbps)	Calculated path	Priority
ETP1_Flow1	4000	4000	ETP1-path1	6
ETP1_Flow2	10000	10000	ETP1-path1	5
ETP1_Flow3	10000	10000	ETP1-path4	4
ETP1_Flow4	9666	10000	ETP1-path4	3
ETP1_Flow5	3222	30000	ETP1-path1	1
ETP2_Flow1	4000	4000	ETP2-path1	6
ETP2_Flow2	16111	20000	ETP2-path2	5
ETP2_Flow3	10000	10000	ETP2-path3	4
ETP2_Flow4	9666	10000	ETP2-path3	3
ETP2_Flow5	3222	30000	ETP2-path2	1

Table 5: Continued.

Flow name	Calculated bandwidth (kbps)	Demand bandwidth (kbps)	Calculated path	Priority
ETP3_Flow1	4000	4000	ETP3-path1	6
ETP3_Flow2	6000	10000	ETP3-path3	5
ETP3_Flow3	5000	10000	ETP3-path2	4
ETP3_Flow4	3750	10000	ETP3-path4	3
ETP3_Flow5	1250	30000	ETP3-path4	1

Table 5 increases the required bandwidth of enterprise 2 application stream 2 compared with Table 4. From the calculation results, it can be seen that the allocated bandwidth of application flow under the same conditions of enterprise 1 and enterprise 2 is consistent, which shows the global fairness of the algorithm.

4. Conclusions and Outlook

Based on the actual network situation of the current enterprise network, this paper introduces a complete solution for SDN service traffic scheduling, including network visualization, traffic scheduling, and configuration distribution. The rationality and correctness of the scheduling algorithm are verified by tests. The research content of this paper is more based on the status quo of the enterprise network to achieve business traffic scheduling. However, with the emergence of new technologies and the upgrading of enterprise equipment, enterprises can also achieve intelligent scheduling of business traffic in a more convenient and intelligent way, improve the bandwidth utilization of private lines, reduce private line costs, and improve operational efficiency. The method to improve the accuracy of network indicators, the ubiquity of scheduling algorithms, and the efficiency of configuration distribution are also the key directions of future research and practice and ultimately realize the complete integration of services and networks. The dynamic scheduling and universality of future data in the research of this paper are one of the key tasks of future research.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] L. M. Surhone, M. T. Tennoe, and S. F. Henssonow, *Simple Network Management Protocol*, Springer, Berlin Heidelberg, 2006.

[2] F. Chen and B. Xiaoxue, "SDN network topology generation system based on bgp-ls protocol [J]," *Network new media technology*, vol. 5, no. 1, pp. 21–26, 2020.

[3] S. Dong, H. Mo, and J. Hu, "Probe based network condition monitoring technology," *Communication technology*, vol. 52, no. 4, pp. 908–911, 2019.

[4] F. L. . Faucheur, "Requirements for support of differentiated services-aware MPLS traffic engineering," *British Journal of Plastic Surgery*, vol. 2, 2003.

[5] T. Yuan, *Research on Node Migration and Optimization Method for SDN Transition [D]*, Beijing University of Posts and telecommunications, Beijing, China, 2018.

[6] S. Jain, A. Kumar, S. Mandal et al., "B4: experience with a globally-deployed software defined WAN," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM*, pp. 3–14, Hongkong China, August 2013.

[7] E. Danna, A. Hassidim, and H. Kaplan, "Upward max min fairness," in *Proceedings of the INFOCOM, 2012 Proceedings IEEE*, pp. 837–845, IEEE, USA, March 2012.

[8] I. Seremet and S. Causevic, "An analysis of reconvergence delay when using BGP-LS/PCEP as southbound protocols [C]," in *Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics*, MIPRO), Opatija, Croatia, May 2019.

[9] H. Sun, "Optimization model of Internet traffic collection based on NetFlow," *Data communication*, vol. 4, no. 6, pp. 11-12, 2019.

[10] W. Jiahui, *Research and implementation of network traffic analysis*, North China Electric Power University (Beijing), Beijing, China, 2018.

[11] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics*, vol. 27, no. 4, pp. 526–530, 1970.

[12] G. Trimponias, Y. Xiao, H. Xu, X. Xu, and Y. Geng, *Centrality-based Middlepoint Selection for Traffic Engineering with Segment Routing*, USA, 2017.

[13] R. Enns, M. Bjorklund, and J. Schoenwaelder, *Network Configuration Protocol (NETCONF)*, USA, 2011.

[14] K. Nichols, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC-2474*, USA, 1998.