Multi-Service Provider Network Traffic Scheduling Method Based on Application Perception

Zhenyu Wu¹, Pei Zhang*, Xiaohong Huang¹ and Qun Cong²

¹Network Information Center, Beijing University of Posts and Telecommunications, Beijing, 100876, China
²Beijing WRD Technology Co., Ltd., Beijing, 100876, China
*Corresponding author's e-mail: zhangpei@bupt.edu.cn

Abstract. Multi-service provider network traffic scheduling is an important research topic. Previous scheduling methods based on DNS resolves the issues of high latency and package loss due to cross-ISP access. However, there are still some problems like large bandwidth fluctuations and low bandwidth utilization in existing methods due to low scheduling accuracy. In this article, we propose a scheduling method based on application perception to solve these issues. Our scheduling method consists three components: application perception, traffic prediction of uplinks and network traffic scheduling based on DNS. It performs scheduling for each elephant flow application to improve scheduling accuracy. The experiment result shows that the application perception based method improves scheduling performance from both scheduling stability and bandwidth utilization.

1. Introduction

In order to improve the network access speed and prevent the entire campus network from crashing due to an uplink failure, more and more campus network builders choose to set up multiple ISP (Internet Service Provider) uplinks [1], and multi-service provider network traffic scheduling has become an important research topic.

Many campus network builders choose to deploy load balancing devices at the network egress [2]. The core of the load balancer is to use SNAT to translate the source IP address and then forward it to the corresponding ISP uplink [3]. However, due to the low quality of the network connection between domestic ISPs (especially in China), the problems of high latency of users' cross-ISP access and the loss of packets still exist [2][4].

In recent years, with the popularity of CDN (Content Distribution Network) technology, Internet applications have begun to provide related services on multiple ISP networks. Many researchers have used CDN and DNS (Domain Name System) technology [5] to optimize multi-service provider network traffic scheduling. In [6], the intranet users are IP-grouped, and the DNS requests of each group user are forwarded from the DNS resolution node in the campus to the DNS resolver of the corresponding ISP. Due to the existence of CDN technology, the destination IP of each packet belongs to each ISP. Therefore, only destination address routing is needed at campus network egress to achieve traffic

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Published under licence by IOP Publishing Ltd
scheduling. However, the static partitioning method does not have high reliability, and cannot cope with problems such as single uplink failure and traffic burst in single IP group. The method proposed in [7] solves the above problem by parsing the DNS requests in the campus according to the real-time idle value of each uplink, but this method still has defects: the actual traffic ratio between applications is usually not 1:1. Hence the actual scheduling result has a certain error compared with the prediction result, and it is prone to the phenomenon that the scheduling effect is not obvious or the scheduling is excessive. In the peak traffic period, there may even be a congestion-idle phenomenon: there is an uplink being congested and then being idle immediately and then congested.

Aiming at the problems in the above scheduling strategies, a multi-service provider network traffic scheduling method is proposed based on application perception and [7]. The application perception is added to the traffic scheduling to enable the DNS to schedule for different applications, thereby large-flow applications’ traffic can be better split into multiple uplinks, scheduling accuracy and smoothness can be improved.

The organizational structure of this article is as follows: Section 1 introduces the current research status of the subject and the basic principles of multi-service provider network traffic scheduling method based on application perception. Section 2 elaborates on the basic model of the algorithm. The actual experiment in Section 3 verifies the feasibility and superiority of the algorithm. Section 4 summarizes the above algorithms and experimental results.

2. Traffic Scheduling Method

The multi-service provider network traffic scheduling algorithm based on application perception consists of three components: application perception, traffic prediction of uplinks and network traffic scheduling.

2.1 Application perception

The application perception is used to extract applications that need to perform traffic scheduling in the campus egress network traffic. Since it is very difficult to schedule network traffic for each application under a large network scale, we extract only the elephant flow applications in view of the elephants and mice phenomenon [8]. The phenomenon of elephant flow and mouse flow is that most of the data traffic in the export link is generated by a small number of applications. This small part of the application is for elephant flow application. Take Beijing University of Posts and Telecommunications as an example, streaming media (video live, music) and resource downloads (cloud disk, application stores) account for 70% to 80% of the bandwidth.

Here we use PANABIT [9][10] gateway to help analysis application protocol from network traffic, and it will inserts the analysis result into the NAT log. The log body mainly includes the session start/end time, upstream traffic bytes, downstream traffic bytes, application protocols. We extract the elephant flow application by performing periodic statistical analysis on the NAT log. Assuming a time series \( t_1, t_2, t_3, \ldots, t_n \), the bandwidth of uplink \( k \) is \( b(t_i, k) \) at time \( t_i \), and the bandwidth occupied by application \( m \) in uplink \( k \) is \( a(t_i, m, k) \). After performing statistical analysis on the NAT log, the application is sorted in descending order by the occupied bandwidth. Then we take the top \( n \) effective elephant flow application, the value \( n \) satisfies the following formula:

\[
\sum_{m=1}^{n} \sum_{k=1}^{u} a(t_i, m, k) \geq r \sum_{k=1}^{u} b(t_i, k), r \in (0,1)
\]  

In this formula, \( u \) represents the total number of uplinks, and \( r \) is the judgment threshold for the application of the elephant flow.

After obtaining the applications of the elephant flow, we collect the domain name set of these applications by disassembling the application data package and crawling from websites. Algorithm 1 summarizes the basic algorithm flow of this model.
2.2 Traffic prediction of uplinks

The traffic prediction component is used to sense the bandwidth utilization of each uplink in advance, and avoids single uplink congestion during peak traffic periods by predicting and scheduling in advance. The prediction model in [7] which based on the Holt-Winters [12][13] does work well, the average prediction error is about 3%. Here we use the ARIMA (self-regressive integral moving average) model [14][15] for prediction of each uplink bandwidth. Compared with the prediction model in [7], it has higher prediction accuracy. The process formula is as follows:

\[(1 - \sum_{i=1}^{P} \phi_i L^i)(1 - L)^d X_t = \delta + (1 + \sum_{i=1}^{P} \theta_i L^i)\epsilon_t\]  \((2)\)

Where \(L\) represents the lag operator, \(\phi_i\) is the parameter of the autoregressive process AR, \(\theta_i\) is the parameter of the moving average process MA, and \(\epsilon\) represents the prediction error term. The establishment of ARIMA prediction model is divided into four steps: (1) to detect the stationarity of time series data, here we use ADF test [16] to determine. (2) If the time series is not stationary, perform differential operation on the data, and then repeat the previous operation; if the time series is stationary, proceed to the next step. (3) Use the ACF (autocorrelation function) and the PACF (partial autocorrelation function) to explore the reasonable range of the parameter \(p\) and parameter \(q\) in the model, and then use the AIC to optimize these parameters. (4) Use the \(p, d, q\) obtained in the above steps to test the ARIMA\((p,d,q)\) model. In this paper, the forecast data is compared with the actual data, and the average absolute error is measured to measure the accuracy of the prediction model. The formula is defined as follows:

\[MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|p_i - R_t|}{R_t}\]  \((3)\)

Where \(p_i\) and \(R_t\) represent the predicted and actual values of the flow at time \(t\), respectively. Algorithm 2 gives the flow of the prediction algorithm for the uplink \(k\) traffic time series.

| Algorithm 1 application perception | Algorithm 2: Traffic prediction |
|-----------------------------------|--------------------------------|
| **Input:** t1, t2 (Detect start time and end time) | **Input:** Bandwidth(t1, k) |
| **Output:** DomainList(t1, t2) | **Output:** Bandwidth(t2, k) |
| **Ensure:** sum := 0; threshold 1: applications := | **1:** while |
| DetectApplications() | ADFTest(bandwidth(t1,k)) |
| 2: | Failed |
| SortByBandwidth(applications) | 2: Differentiate |
| 3: foreach applications as app | bandwidth(t1,k) |
| 4: sum := sum + bandwidth(app) | 3: end while |
| 5: if sum <= threshold then | 4: GetACF() && GetPACF() |
| 6: Collect domains of app | 5: p, q := BuildARIMA() |
| 7: Add domains to DomainList(t1, t2) | 6: OptimizeByAIC(p,q) |
| 8: else | 7: bandwidth(t2,k) := |
| 9: goto 11 | ARIMA(p,d,q) |
| 10: end if | 11: end foreach |
2.3 Traffic scheduling

In the 2.1, we extracted the elephant flow application. For these elephant flow application, the scheduling method performs traffic scheduling separately. For a large number of mouse flow applications, we treat them as one elephant flow application. At time \( t_j \), the probability formula for applying \( m \) to the egress uplink \( k \) is:

\[
p(k | m) = \frac{\text{free}(k)}{\sum_{j=1}^{u} \text{free}(j)}
\]

Where \( \text{free}(k) \) represents the predicted idle bandwidth value for uplink \( k \) and \( u \) represents the total number of uplinks.

After the probability is obtained, the DNS server allocates the domain name corresponding to the scheduling application to each link for resolution. We use the link weight wheel to realize the link assignment of the application domain name. Each elephant flow application domain name corresponds to one weight wheel, and all mouse flow applications apply a common weight wheel. As shown in Figure 1, the slots in the weight wheel respectively correspond to one parsing link, and the ratio of the number of slots allocated to each link satisfies the probability distribution of Formula 4.

![Figure 1. Link weight wheel. The pointer in the weight wheel points to the slot corresponding to the current parsing uplink and it changes by rotating the pointer.](image)

The flow scheduling algorithm flow is as shown in Algorithm 3. The calculation of the link weight wheel is a periodic event, and runs in parallel with the DNS split link resolution.

```
Algorithm 3 Traffic scheduling

Input: DomainList(t), AvailableBandwidthList

Output: NULL

THREAD 1
1: while Sleep(t) do
2:   foreach DomainList(t) as domain do
3:     CreateQueryNameWeightWheels From(domain, AvailableBandwidthList)
4:   end foreach
5:   CreateCommonNameWeightWheel From(AvailableBandwidthList)
6: end while

THREAD 2
1: domain := AcceptRequest()
2: if domain in DomainList(t) then
3:   GetWeightWheelFromWheels(domain)
4: else
5:   GetCommonWheel()
6: end if
7: link := GetServiceProviderFromWeight()
8: WheelMovePointForward()
9: DNSQueryFromLink(link)
```
In order to quantitatively analyze the volatility of the scheduling effect, we use the uplink average fluctuation value $AD$ to measure the stability of the scheduling effect. The formula of $AD$ is defined as follows:

$$AD = \frac{1}{u} \sum_{k=1}^{u} \sigma(\nabla b(t, k))$$  \hspace{1cm} (5)

Where $\nabla b(t, k)$ represents the first-order difference of the time series data $b(t, k)$ of the uplink $k$, the purpose of the difference is to filter out the effects of the total bandwidth fluctuation, and $\sigma$ represents the standard deviation formula.

3. Experiment

In order to verify that the scheduling method has better scheduling effect, we built the relevant network equipment according to Figure 2 on the campus network of Beijing University of Posts and Telecommunications and conducted comparative experiments. The topology ISPs are China Telecom and China Edu. Network. The campus inner network is a typical NAT network and the DNS resolver IP addresses of campus users are dispatched by DHCP servers.

The experimental framework is mainly divided into four layers: (1) The uplink layer contains the egress route and ISP uplinks, which is the lowest layer of the framework. (2) The data acquisition layer collects the bandwidth usage of each uplink and the bandwidth of applications in the traffic. (3) The analysis scheduling layer performs bandwidth prediction on each uplink, filters the elephant flow application in the collected data, and obtains the domain names corresponding to the applications, and then calculates the pre-scheduling policy. (4) The top-level DNS scheduling layer executes the application scheduling policy output by the analysis scheduling layer.

The application data collection period is 5 minutes. After the data is aggregated, the effective elephant flow application is extracted according to Formula 1. The threshold value is 0.8 in this experiment. Figure 3 shows the application bandwidth and egress uplink bandwidth data of the Beijing University of Posts and Telecommunications for a period of time. After obtaining the valid elephant flow application, the domain name corresponding to the application needs to be further extracted. For the HTTP protocol application, the domain name information can be obtained after the data packet is
disassembled in the HTTP protocol packet, and for other types of applications, it is obtained by manually checking and using the crawler tool.

The collection of each egress uplink bandwidth utilization reference [7], the acquisition cycle time is 5 minutes. In order to match the traffic prediction, we use the ARIMA prediction algorithm in 2.2 to predict the bandwidth value of the last 5 minutes, and calculate the prediction MAPE for this predicted value in the next acquisition period. Figure 4 shows the MAPE over a period of time. The prediction model has a good prediction effect in this environment, and the average MAPE value is about 0.3%.

We conducted a related comparison experiment with the prediction algorithm in [7]. Figure 5 shows the average prediction MAPE values of the two prediction models at different prediction times. The data shows that the model has higher accuracy than the Holt-Winters based model when the prediction duration is within 50 minutes.

In the traffic scheduling part, this experiment is based on the open source DNS software BIND [17] [18]. we added the link weight wheel function to BIND. The traffic scheduling period is 5 minutes, the scheduler sends a batch processing instruction to the DNS software, and the DNS software allocates a link weight wheel for the corresponding domain name according to the instruction. In order to compare with the scheduling effect in [7], we added the scheduling function of [7] on BIND too. In the process of setting up the control group, both of them use the prediction algorithm with higher accuracy to predict. The experiment time is set to two adjacent working days. Finally, we selected the data with a total traffic difference of less than 3% as a comparison. Theoretically, the total traffic without bandwidth limitation should be the same in this experiment. Here it is too difficult to find the same two pieces of data under actual test conditions, and the 3% deviation rate is acceptable for fluctuation AD comparison. Figure 6 shows the traffic scheduling effect of the scheduling method and the scheduling method in [7] for a period of time. It can be seen from the figure that during this time, the scheduling effect of the method is more stable, and the bandwidth of each link is more volatile under the scheduling in [7].
Figure 6. Scheduling result comparison.

Table 1 shows the $AD$ comparison data between our method and method in [7]. We also selected experimental data with a total traffic difference of less than 3%. The data shows that during the peak period between 9:00 and 10:00 pm, the fluctuation value $AD$ decreased by 33.4%, within 24 hours the $AD$ value decreased by 19.50%. Compared with the method in [7], our method has greatly improved the stability of the scheduling effect.

| Time period       | $AD$ of method in [7] | $AD$ of ours |
|-------------------|-----------------------|--------------|
| 10:00–11:00       | 76.12                 | 72.65        |
| 14:00–15:00       | 42.37                 | 37.74        |
| 21:00–22:00       | 95.57                 | 63.56        |
| 00:00–24:00       | 67.91                 | 54.66        |

Furthermore, we conducted a comparative experiment to compare the total bandwidth utilization of the two during peak hours. The egress bandwidth of each uplink is limited to 1Gbps in this process. Table 2 shows the bandwidth utilization comparison data. The total traffic deviation rate here is set to within 1%, which is stricter than $AD$ comparison experiment. As can be seen from the table, by using our scheduling method the bandwidth utilization can be increase to 84.99%, which is 8.43% higher than the method in [7].

| Method in [7] | Ours           |
|---------------|----------------|
| Average utilization (%) | 76.56 | 84.99 |
| Total traffic (Mbyte) without bandwidth limitation | 1796550.38 | 1785373.01 |

4. Conclusion
The DNS-based multi-service provider traffic scheduling method solves the problems of high latency and packet loss caused by cross-ISP access. However, the existing DNS-based multi-service provider...
network traffic scheduling method has large bandwidth fluctuations and low bandwidth utilization due to low scheduling accuracy. This article proposes a scheduling method based on application perception, which optimizes the existing methods from the following two aspects: (1) Adding an application perception algorithm to separately schedule the elephant flow application. (2) Use the ARIMA model to predict the uplink traffic and improve the prediction accuracy. Experimental data shows that the application perception based traffic scheduling method has a satisfying performance from both scheduling stability and bandwidth utilization.

Acknowledgment
Foundation Items: Research Fund of Ministry of Education - China Mobile under Grant No. MCM20160304

References
[1] C. Huang. (2013) A Successful Design on Load Balancing Method Based on Campus Network Program. Southwest Jiaotong University Master Degree Thesis.
[2] H. Zhang, R. Zhou, X. Zhang. (2012) Design and Realization of Multi-Export Campus Network Based on Redundancy Architecture. Computer Science, 2012,39(S2): 219-222.
[3] H. Wang, J. Zhang. (2004) Link Load Balancing Strategy in Multi-Link. Acta Scientiarun Naturaltium Universitatis Nankaiensis, 2004(04): 59-63.
[4] Y. Gao. (2014) The Research and Design of Campus Network’s Gateway Upgrade Program. YunNan University Master Degree Thesis.
[5] Z. Qin, F. Zhou, L. Li. (2013) The Impact of DNS on CDN Streaming Performance. Journal of University of Electronic Science and Technology of China, 2013,42(04): 577-580.
[6] Q. Shan, F. Nan. (2017) Application of DNS Multi-Cache Policy in Multi-Outlet Traffic Optimization. Modern Computer, 2017(29): 39-43.
[7] Y. Weng, X. Huang, D. Li. (2017) Network multi-export traffic scheduling method based on DNS. Journal of Southeast University (Natural Science Edition), 2017,47(S1): 102-107.
[8] Papagiannaki, Konstantina. (2002) A pragmatic definition of elephants in internet backbone traffic. Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement. ACM.
[9] L. Liu, W. Li, M. Xiao. (2008) Management Policy and Traffic Monitor of P2P on Campus Network. Journal of Kunming University of Science and Technology (Science and Technology), 2008(03):45-48.
[10] G. Hou, J. Wang. (2016) Research of PANABIT in Campus Network. Science & Technology Information, 14(34):37+39.
[11] H. Jiang, Z. Yang. (2012) Research and Application on Improved Index Algorithm of Mass NAT Log Search. Computer Science, 39(S2):191-194.
[12] X Zhang. (2006) The Investigation of Exponential Smoothing. Journal of Inner Mongolia Agricultural University (Natural Science Edition), 2006(04): 153-156.
[13] Carl de Boor. (2001) A Practical Guide to Splines. Springer, Germany.
[14] J. Cao, et al. (2017) Towards tenant demand-aware bandwidth allocation strategy in cloud datacenter, Future Generation Computer Systems. 2017.
[15] George E. P. B. (2015) Time Series Analysis: Forecasting And Control. John Wiley Sons, San Francisco.
[16] Allan W. G., Bruce E. H. (1996) Residual-based tests for cointegration in models with regime shifts. Journal of Econometrics, 1996, 70(1): 99-123.
[17] C. Li, J. Yu. (2017) Campus network building intelligence DNS based on BIND9. Information Technology, 2017(06): 91-95.
[18] Cricket Liu, Paul Albitz. (2006) DNS and BIND. O'Reilly Media, California.