IEICE TRANS. INF. & SYST., VOL.E99–D, NO.9 SEPTEMBER 2016

LETTER

Tardy Flow Scheduling in Data Center Networks

Gyuyeong KIM[†], *Nonmember and* **Wonjun LEE**[†a)], *Member*

SUMMARY Query response times are critical for cluster computing applications in data centers. In this letter, we argue that to optimize the network performance, we should consider the latency of the flows suffered loss, which are called tardy flows. We propose two tardy flow scheduling algorithms and show that our work offers significant performance gains through performance analysis and simulations.

key words: *compute clusters, data center networks, flow completion time, flow scheduling*

1. Introduction

Data centers are burgeoned owing to the migration of interactive cluster computing applications having a penchant for low latency (e.g. big data analytics frameworks) [1]. These applications follow a workflow that parallelizes a requested user query into multiple task flows. The query cannot be completed until all flows have finished, so the flow completion time (FCT) of the slowest flow is crucial to query response times.

For the low FCT, precursors decrease the queueing delay by dispersing network traffic [2] and reducing sending rates [3]. Although these mechanisms work well in the absence of loss, flows still experience the high FCT when packet loss occurs. For short-lived flows, suffering loss often indicates the retransmission of *entire* flows. Unfortunately, data centers are prone to network component failures (e.g. links, racks, and switches). A measurement study [4] finds that roughly 1.5 influential failures having long downtime occur every hour. At this moment, losses are inevitable and even a single failure can lead to multiple losses. Furthermore, the shallow buffered and shared-memory nature of data center switches [3] exacerbates the problem. Flows easily face loss because switch buffers have no rooms to absorb packet bursts entirely. Then, how can we minimize query response times *in the presence of loss?*

In this letter, we argue that the FCT of tardy flows plays a key role to determine query response times. The *tardy flow* is defined as the flow experienced loss. A tardy flow is often the slowest flow in a query because retransmissions cause the additional delay. Therefore, we try to minimize the FCT of tardy flows by designing two tardy flow scheduling algorithms, Tardy Flow Scheduling (TFS) and TFS+. These algorithms prioritize tardy flows over normal flows across the data center fabric. Through performance analysis and ns-3 simulations, we demonstrate that our work provides significant performance gains by up to 2.25× and 2.01× for the mean and the 99.9th percentile response time of queries, respectively.

Manuscript received February 5, 2016.
Manuscript revised May 11, 2016.
Manuscript publicized May 25, 2016.

[†]The authors are with the Network Research Lab., Graduate School of Information Security, Korea University, Seoul, Republic of Korea.
a) E-mail: wlee@korea.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2016EDL8038

2. Network Model

Consider a flow-level network model where a data center network is abstracted as a giant non-blocking switch that interconnects all servers and has a single M/G/1/First-Come-First-Serve (FCFS) queue with infinite buffer. The input and output ports are at the network interface card of servers. Since common data center topologies consist of multi tiers of switches (e.g. fat-tree [5]), input and output ports can be seen as uplinks and downlinks, respectively. We assume a Poisson arrival process with mean arrival rate λ having size $S \sim F(\cdot)$ where $F(\cdot)$ is the CDF.

Let $\mathbb{E}[Q]$ represent the mean queueing delay for a M/G/1/FCFS queue. We assume that each flow sees the mean queueing delay due to lack of a closed form solution for M/G/1 queueing delay distributions. $\mathbb{E}[Q]$ is easily derived from the Pollaczek-Khinchine formula [6] that $\mathbb{E}[Q] = \frac{\rho}{1-\rho} \cdot \mathbb{E}[S_e]$ where $\mathbb{E}[S_e] = \frac{\mathbb{E}[S](C^2+1)}{2}$ and $\rho = \lambda\mathbb{E}[S] \in [0, 1)$ denotes traffic load, $\mathbb{E}[S]$ means the mean flow size, and $C^2 = \mathrm{VAR}(S)/\mathbb{E}[S]^2$ indicates the squared coefficient of variation of S, simply the flow size variability.

Now, let us consider tardy flows. We define tardy flows as the flows experienced loss. The retransmission of tardy flows increases the overall traffic load ρ to $\frac{\rho}{1-\omega}$ where ω denotes the loss probability and $\frac{1}{(1-\omega)} = \sum_{i=1}^{\infty} \omega^{i-1}$ represents the additional traffic load for repetitive loss and the corresponding retransmission. Then, we have the mean queueing delay considering tardy flows for a M/G/1/FCFS queue by rewriting $\mathbb{E}[Q]$ as $\mathbb{E}[Q'] = \frac{\frac{\rho}{1-\omega}}{1-\frac{\rho}{1-\omega}} \cdot \mathbb{E}[S_e] = \frac{\rho}{(1-\omega)-\rho} \cdot \mathbb{E}[S_e]$. We find that $\mathbb{E}[Q']$ is sensitive to ρ, ω, and flow size distributions. It is also easy to see that $\mathbb{E}[Q'] = \mathbb{E}[Q]$ for $\omega = 0$.

Define $\mathbb{E}[FCT_s^{\mathrm{Norm}}]$ as the expected FCT of a flow having size s without loss (i.e. a normal flow). Since our model considers flow level scheduling with a single queue, we use service delay and queueing delay to represent the expected FCT. Then, we have:

Algorithm 1 Tardy Flow Scheduling

```
Require: ⟨Flow⟩F                                  ▷ Active flows at a time
1: ⟨Queue⟩H, L ← φ                                ▷ Virtual queues
2: for all f ∈ F do
3:     if f_tardy = TRUE then
4:         H.add(f)                               ▷ Tardy flows
5:     else L.add(f)                              ▷ Normal flows
6:     end if
7: end for
8: ⟨Queue⟩Q ← H ∪ L                               ▷ Set of flows to schedule at a time
9: return Q.Front()
```

$$\mathbb{E}[FCT_s^{\text{Norm}}] = \frac{s}{\mu} + \mathbb{E}[Q'] \tag{1}$$

where s/μ is the service delay. We assume service rate $\mu = 1$ for the sake of tractability.

Let $\mathbb{E}[\mathbb{E}[FCT]]$ be the mean FCT of normal flows in the network. Simply, we use $\mathbb{E}[FCT]$ to indicate $\mathbb{E}[\mathbb{E}[FCT]]$. $\mathbb{E}[FCT]$ is given by:

$$\mathbb{E}[FCT] = \int_0^\infty \left(\frac{s}{\mu} + \mathbb{E}[Q']\right) \frac{f(s)}{F(\infty)} ds \tag{2}$$

where $f(\cdot)$ is the PDF.

The expected FCT of a tardy flow having size s is:

$$\mathbb{E}[FCT_s^{\text{Tardy}}] = \frac{s}{\mu} + \mathbb{E}[Q'] + n\mathbb{E}[Q''] \tag{3}$$

where n is the number of retransmissions and $\mathbb{E}[Q'']$ denotes the retransmission delay.

In this letter, we consider an objective for optimizing query response times in the presence of loss: *minimizing the FCT of tardy flows*. In $\mathbb{E}[FCT_s^{\text{Tardy}}]$, $\mathbb{E}[Q']$ is the delay for initial transmission. Since a flow is regarded as a tardy flow after experiencing loss, $\mathbb{E}[Q'']$ is our primary concern in flow scheduling. The optimal FCT of a tardy flow can be achieved by making $\mathbb{E}[Q''] = 0$. Thus, our work strives to minimize $\mathbb{E}[Q'']$.

3. Tardy Flow Scheduling

We introduce TFS and TFS+, two tardy flow scheduling algorithms to accomplish the objective.

TFS: TFS is a simple yet effective flow scheduling algorithm having favoritism toward tardy flows (Algorithm 1). Basically, like a general flow scheduling algorithm, TFS schedules flows in a FCFS fashion. However, tardy flows are prioritized over normal flows, so normal flows can be scheduled only if there exist no competing tardy flows. We can express the difference of delay that tardy/normal flows see by using virtual queue H and L. The overall arrival rate $\frac{\lambda}{1-\omega}$ is divided up between queue H and L. Tardy flows are enqueued in queue H while the other flows go to queue L. Therefore, queue H and L are with $\lambda_H^{\text{TFS}} = \frac{\omega\lambda}{1-\omega}$ and $\lambda_L^{\text{TFS}} = \lambda$, respectively. The mean queueing delay of the flow with priority k can be easily derived from $\mathbb{E}[Q]$ and is known as $\mathbb{E}[S_k]/(1 - \sum_{i=1}^{k-1}\rho_i) + (\frac{\lambda}{2}\sum_{i=1}^{k} p_i\mathbb{E}[S_i^2])/(1 - \sum_{i=1}^{k-1}\rho_i)(1-\sum_{i=1}^{k}\rho_i)$ where p_i is the fraction of priority k and $\rho_k = \lambda_k\mathbb{E}[S_k]$ [6]. From the formula, we can derive the mean queueing delay of queue H and L in TFS that $\mathbb{E}[Q_H^{\text{TFS}}] = \mathbb{E}[S] + \frac{\frac{\omega\lambda}{2(1-\omega)}\mathbb{E}[S^2]}{1-\frac{\omega\rho}{1-\omega}}$ and $\mathbb{E}[Q_L^{\text{TFS}}] = \frac{\mathbb{E}[S]}{1-\frac{\omega\rho}{1-\omega}} + \frac{\frac{\omega\lambda}{2(1-\omega)}\mathbb{E}[S^2]}{(1-\frac{\omega\rho}{1-\omega})(1-\frac{\omega\rho}{1-\omega}-\rho)}$. We expect that $\mathbb{E}[Q_H^{\text{TFS}}] < \mathbb{E}[Q_L^{\text{TFS}}]$ in general cases because enqueued flows in queue L can be serviced only if queue H is empty and can be preempted when a new flow arrives in queue H.

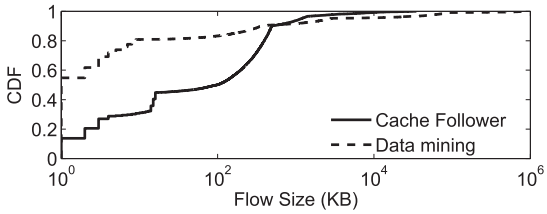


Fig. 1 Two data center workloads used in analysis. These workloads have heavy-tailed flow size distributions where most of flows are short flows.

Consider a toy example. Suppose that we have a number of active flows f. All flows see $\mathbb{E}[Q_L^{\text{TFS}}]$. Now let $\{f_1, f_2\}$ be a set of tardy flows. With TFS, f_1 and f_2 experience $\mathbb{E}[Q''] = \mathbb{E}[Q_H^{\text{TFS}}]$ for retransmissions. The other flows bear the additional delay of the tardy flows. One potential cost of TFS is the increase in the FCT for normal flows. Thus, we examine the benefit and cost of TFS in the next section.

TFS+: TFS overlooks one important point of data center traffic, heavy-tailed distributions. Figure 1 shows the overall flow size distributions of a cache follower of web servers [7] and a data mining application [8]. The both workloads are heavy-tailed. For example, approximately 80% of flows are smaller than 100KB in the data mining workload because queries generally only call for the execution of tasks. Large flows are for network update and machine configuration jobs, not of user queries. In this context, expediting large tardy flows is redundant because of head-of-line blocking.

TFS+ is an improvement of TFS that concerns only short tardy flows. TFS+ works as similar to TFS in Algorithm 1, but excludes large tardy flows. The condition 'f_{tardy} = TRUE' at line 3 becomes 'f_{tardy} = TRUE and $f_{\text{size}} \le S_L$' in TFS+ where f_{size} is the flow size and S_L denotes a given threshold of large flows. Thus, flows arrive with rate $\lambda_H^{\text{TFS+}} = \omega\kappa\lambda/(1-\omega)$ where $\kappa = \int_0^{S_L} f(s)ds$ is the fraction of the number of flows less than S_L. Queue L is with $\lambda_L^{\text{TFS+}} = (1-\omega\kappa)\lambda/(1-\omega)$. Using the arrival rates, we obatin the mean queueing delay for queue H and L in TFS+. For queue H, $\mathbb{E}[Q_H^{\text{TFS+}}] = \mathbb{E}[S]_{\le S_L} + \frac{\frac{\omega\lambda}{2(1-\omega)}\mathbb{E}[S^2]_{\le S_L}}{1-\rho_H^{\text{TFS+}}}$ where $\mathbb{E}[S]_{\le S_L}$ denotes the mean size of flows less than S_L and $\mathbb{E}[S^2]_{\le S_L} = \text{VAR}(S) + \mathbb{E}[S]^2_{\le S_L}$ is the second moment of S. Similarly, for queue L, we have that $\mathbb{E}[Q_L^{\text{TFS+}}] = \frac{\mathbb{E}[S]_{>S_L}}{1-\rho_H^{\text{TFS+}}} + \frac{\frac{\omega\lambda}{2(1-\omega)}\mathbb{E}[S^2]}{(1-\rho_H^{\text{TFS+}})(1-\rho_H^{\text{TFS+}}-\rho_L^{\text{TFS+}})}$ where $\mathbb{E}[S]_{>S_L}$ is the mean flow size larger than S_L and $\rho_H^{\text{TFS+}} = \frac{\omega\kappa\lambda\mathbb{E}[S]_{\le S_L}}{1-\omega}$, $\rho_L^{\text{TFS+}} = \frac{(1-\omega\kappa)\lambda\mathbb{E}[S]_{\text{TFS+},L}}{1-\omega}$ where $\mathbb{E}[S]_{\text{TFS+},L} =$

$\frac{(1-\omega)\mathbb{E}[S]_{\mathrm{TFS}+}+\omega(1-\kappa)\mathbb{E}[S]_{>S_L}}{(1-\omega)+\omega(1-\kappa)}$ and $\mathbb{E}[S]_{\mathrm{TFS}+} = \omega\kappa\mathbb{E}[S]_{\leq S_L} + ((1-\omega)\,\mathbb{E}[S] + \omega\,(1-\kappa)\,\mathbb{E}[S]_{>S_L})$. Note that $\mathbb{E}[S]_{\mathrm{TFS}+} = \sum_i^n p_i\mathbb{E}[S_i]$.

In the aforementioned example, suppose that f_1 is a large tardy flow and f_2 is a short tardy flow. Then, f_2 sees $\mathbb{E}[Q_H^{\mathrm{TFS}+}]$ for $\mathbb{E}[Q'']$, but $\mathbb{E}[Q''] = \mathbb{E}[Q_L^{\mathrm{TFS}+}]$ for f_1.

4. Performance Analysis

We analyze the performance of TFS and TFS+. As we mentioned, flows consisting of queries are short flows. Therefore, we focus on short flows in analysis.

We obtain the mean FCT of short normal flows for TFS using (2) as that:

$$\mathbb{E}[FCT_{\mathrm{TFS}}^{\mathrm{Norm}}] = \int_0^{S_L}\left(\frac{s}{\mu} + \mathbb{E}[Q_L^{\mathrm{TFS}}]\right)\frac{f(s)}{F(S_L)}ds \tag{4}$$

Similarly, the mean FCT of short tardy flows in TFS is:

$$\mathbb{E}[FCT_{\mathrm{TFS}}^{\mathrm{Tardy}}] = \int_0^{S_L}\left(\frac{s}{\mu} + \mathbb{E}[Q_{\mathrm{T}}^{\mathrm{TFS}}]\right)\frac{\frac{\omega}{1-\omega}f(s)}{F(S_L)}ds \tag{5}$$

where $\mathbb{E}[Q_{\mathrm{T}}^{\mathrm{TFS}}] = \mathbb{E}[Q_L^{\mathrm{TFS}}] + n\mathbb{E}[Q_H^{\mathrm{TFS}}]$ and $\frac{\omega}{1-\omega}$ is the fraction of tardy flows in the network.

For TFS+, we obtain the mean FCT of short normal flows as following that:

$$\mathbb{E}[FCT_{\mathrm{TFS}+}^{\mathrm{Norm}}] = \int_0^{S_L}\left(\frac{s}{\mu} + \mathbb{E}[Q_L^{\mathrm{TFS}+}]\right)\frac{(1-\omega\kappa)f(s)}{F(S_L)}ds \tag{6}$$

where $\omega\kappa$ is the fraction of short tardy flows. Therefore, $(1-\omega\kappa)$ is naturally the fraction of flows that are not short tardy flows.

The mean FCT of short tardy flows in TFS+ is given by:

$$\mathbb{E}[FCT_{\mathrm{TFS}+}^{\mathrm{Tardy}}] = \int_0^{S_L}\left(\frac{s}{\mu} + \mathbb{E}[Q_{\mathrm{T}}^{\mathrm{TFS}+}]\right)\frac{\omega\kappa f(s)}{F(S_L)}ds \tag{7}$$

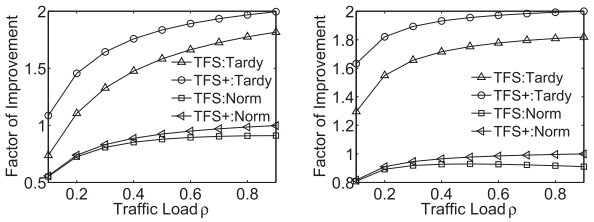
where $\mathbb{E}[Q_{\mathrm{T}}^{\mathrm{TFS}+}] = \mathbb{E}[Q_H^{\mathrm{TFS}+}] + n\mathbb{E}[Q_L^{\mathrm{TFS}+}]$.

We now evaluate our work using numerical results. In the absence of alternatives in the literature, we compare our work with the baseline having no consideration of tardy flows. The mean FCT of short flows for the baseline is: $\mathbb{E}[FCT^{\mathrm{Norm}}] = \int_0^{S_L}\left(\frac{s}{\mu} + \mathbb{E}[Q']\right)\frac{f(s)}{F(S_L)}ds$ and $\mathbb{E}[FCT^{\mathrm{Tardy}}] = \int_0^{S_L}\left(\frac{s}{\mu} + (1+n)\,\mathbb{E}[Q']\right)\frac{f(s)}{F(S_L)}ds$. We use the workloads in Fig. 1. We set $S_L = 68$ (100KB) and $n = 1$ for all results.

Metric Our performance metric in the mean FCT improvement is a normalized metric that

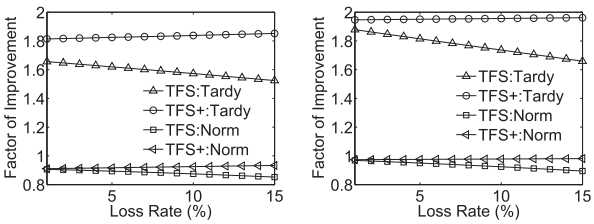
$$\text{Factor of Improvement} = \frac{\mathbb{E}[FCT^{\mathrm{Tardy/Norm}}]}{\mathbb{E}[FCT_{\mathrm{TFS/TFS}+}^{\mathrm{Tardy/Norm}}]}$$

Impact on the mean FCT of flows Fig. 2 shows that tardy flow scheduling improves the performance of tardy flows with slight performance degradation for normal flows. We first find that our work reduces the mean FCT of short tardy flows significantly. We also observe that the algorithms deliver better performance in the data mining workload, which is more heavy-tailed. The factor of improvement increases as the traffic load increases in the all workloads. Especially, TFS+ improves the performance by 1.99× when $\rho = 0.9$ and always offer the performance gain in all cases. The upper bound of the improvement is 2×, which means the retransmission delay of tardy flows is zero. TFS has relatively worse yet satisfying performance where the maximum improvement is 1.82× when $\rho = 0.9$ in Fig. 2 (b). Unfortunately, in the cache follower workload, TFS degrades the performance when $\rho = 0.1$ (Fig. 2 (a)). The gap between TFS+ and TFS stems from that large tardy flows block short tardy flows in TFS.



(a) Cache follower workload (b) Data mining workload

Fig. 2 The improvement in the mean FCT of short flows. $\omega = 0.1$



(a) Cache follower workload (b) Data mining workload

Fig. 3 The impact of loss rate on the mean FCT of short flows.

We now turn our attention to normal flows. We find that the proposed algorithms increase the mean FCT of short normal flows. For example, TFS+ increases the mean FCT by up to 1.21× and mostly zero in the data mining workload. The notable point is the poor performance up to 1.8× when $\rho \leq 0.2$ in Fig. 2 (a). Although this may be significant, however, it is limited when ρ is low. Note that data center networks are often oversubscribed. One possible solution for the problem is to design a scheduling algorithm that changes its policy dynamically with awareness of *the opportunity cost* between the gain and the cost. We leave it to our future work.

Impact of loss rate We now inspect the impact of loss rate on the performance. The increase of loss rate indicates the increase of mean queueing delay that a tardy flow sees. Figure 3 shows that TFS+ has almost the uniform performance regardless of loss rate. This is because TFS+ only prioritizes short tardy flows whose bytes are small. Meanwhile, the performance in TFS is exacerbated as the loss rate in-

creases. The reason is that, unlike TFS+, TFS prioritizes large tardy flows as well. In TFS, short tardy flows are impeded by large tardy flows in front. We say that TFS still has a satisfying performance because the worst performance improvement is 1.53× and the maximum performance deterioration is only 0.82×, respectively. In addition, since the variability of performance is moderate, we say the results in Fig. 2 have consistency with any ω.

5. Packet-Level Simulations

In this section, we conduct a preliminary evaluation of our work using packet-level simulations in ns-3 [9]. Unlike our simplified flow-level network model, this simulation considers all aspects of data center networks.

We simulate a full bisection bandwidth 3-tier fat-tree [5] topology with 54 servers having 10Gbps links and 240μs RTTs on average. Each port in switches has 22.5 KB of shallow buffer size.

We use a realistic workload of production data centers. Queries having size $\{20, 80\}$KB are generated whose arrival rate is set to consume about 10% of the total network throughput. When a server requests a query, 10 randomly chosen servers perform its partitioned tasks. In addition, we make servers generate background traffic having size from 64KB to 32 MB to another randomly selected server representing non-query traffic. We use standard equal cost multiple path (ECMP) routing [2] and TCP NewReno that are widely employed in enterprise data centers [3], [10]. Our settings result in 7% of the packet loss rate and 24% of the tardy flow ratio, respectively.

TFS and the baseline are used for the performance comparison. Switches in TFS prioritizes retransmission packets by using strict priority queueing, thereby reducing the FCT of tardy flows. The baseline schedules all packets in a FCFS manner.

Table 1 shows the mean and the 99.9th percentile query response times. We find that TFS accelerates query response times significantly. Especially, TFS provides the performance gain by 2.25× for 20KB queries. Although the degree is relatively low, our work improves the mean query response times by 1.10× for 80KB queries. The benefit stems from that, as we found in performance analysis, TFS substantially reduces the retransmission delay of tardy flows while increasing the FCT of normal flows slightly. The difference of performance improvement between two queries comes from the query size. As query size goes small, the task flow size also decreases. Then, for a very small flow, packet loss may result in the retransmission of entire flow. Recall that over 50% of flows are 1KB in the data mining workload (Fig. 1). For the 99.9th percentile query response times, TFS provides the performance gain by 2.01× and 1.11× for 20KB and 80KB queries, respectively. The factors of improvement are similar to the result for mean query response times.

Table 1 Mean and the 99.9th percentile query response times (ms) in ns-3 simulations. TFS significantly outperforms the baseline.

Query Size (KB)	Baseline	TFS	Factor of Improvement
20 (Mean)	2.63	1.17	2.25×
80 (Mean)	73.90	67.27	1.10×
20 (99.9th)	165	82	2.01×
80 (99.9th)	334	300	1.11×

6. Conclusion

In this letter, we argued that the FCT of tardy flows has a significant leverage on the query response time for data center applications in the presence of loss. We designed two tardy flow scheduling algorithms called TFS and TFS+ minimizing the retransmission delay of tardy flows. Using flow-level performance analysis and packet-level simulations, we demonstrated that our work offers the substantial performance gain in query response times.

This work is by no means the final word. There remains significant room for improvement in various aspects including opportunity cost-awareness and synthesizing our work with load balancing mechanisms to exploit multi paths in data center networks.

Acknowledgements

This research was sponsored by MSIP, Korea under National Research Foundation grant (No. 2013R1A2A2A01014000).

References

[1] Y. Kiriha and M. Nishihara, "Survey on data center networking technologies," IEICE Transactions on Communications, vol.E96-B, no.3, pp.713–721, March 2013.
[2] M. Shafiee and J. Ghaderi, "A simple congestion-aware algorithm for load balancing in datacenter networks," Proc. IEEE INFOCOM, 2016.
[3] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," Proc. ACM SIGCOMM, pp.63–74, 2010.
[4] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," Proc. ACM SIGCOMM, vol.41, no.4, pp.350–361, 2011.
[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," Proc. ACM SIGCOMM, vol.38, no.4, pp.63–74, 2008.
[6] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, 1st ed., Cambridge University Press, New York, NY, USA, 2013.
[7] A. Roy, H. Zeng, J. Bagga, G. Porter, and A.C. Snoeren, "Inside the social network's (datacenter) network," Proc. ACM SIGCOMM, vol.45, no.5, pp.123–137, 2015.
[8] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," Proc. ACM SIGCOMM, pp.51–62, 2009.
[9] J.L. Font, P. Iñigo, M. Domínguez, J.L. Sevillano, and C. Amaya, "Architecture, design and source code comparison of ns-2 and ns-3 network simulators," Proc. SpringSim, pp.109:1–109:8, 2010.
[10] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," Proc. USENIX NSDI, 2010.