Achieving Application-Level Utility Max-Min Fairness of Bandwidth Allocation in Datacenter Networks

Wangying Ye, Fei Xu(✉), and Wei Zhang

Shanghai Key Laboratory of Multidimensional Information Processing, Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China
fxu@cs.ecnu.edu.cn

Abstract. Providing fair bandwidth allocation for applications is becoming increasingly compelling in cloud datacenters as different applications compete for shared datacenter network resources. Existing solutions mainly provide bandwidth guarantees for virtual machines (VMs) and achieve the fairness of VM bandwidth allocation. However, scant attention has been paid to application bandwidth guarantees for the *fairness of application performance*. In this paper, we introduce a rigorous definition of *application-level utility max-min fairness*, which guides us to develop a non-linear model to investigate the relationship between the fairness of application performance (utility) and the application bandwidth allocation. Based on Newton's method, we further design a simple yet effective algorithm to solve this problem, and evaluate its effectiveness with extensive experiments using OpenFlow in Mininet virtual network environment. Evaluation results show that our algorithm can achieve utility max-min fair share of bandwidth allocation for applications in datacenter networks, yet with an acceptable computational overhead.

Keywords: Bandwidth allocation · Max-min fairness · Application utility · Datacenter networking

1 Introduction

Cloud datacenters are increasingly hosting a variety of big data applications, *e.g.,* MapReduce, Spark, Dryad, transferring a large amount of data between servers [1]. Typically, such applications operate across dozens of servers and initiate a number of heavy network flows over the datacenter networks [2]. Meanwhile, network bandwidth oversubscription is not uncommon in modern datacenters, such as 40 : 1 in some Facebook datacenters [3], which inevitably leads to heavy contention for network resources of core switch bandwidth. Hence, providing fair bandwidth allocation for applications is becoming highly desirable, in order to guarantee the application performance in cloud datacenters [4].

S. Guo et al. (Eds.): CollaborateCom 2015, LNICST 163, pp. 36–46, 2016.
DOI: 10.1007/978-3-319-28910-6_4

However, the traditional TCP rate control mechanisms only provide the *flow-level* max-min fairness [5] or proportional fairness [6] in sharing network bandwidth for applications. Undoubtedly, a "selfish" big data application can request more bandwidth and break such *flow-level* fairness by arbitrarily initiating a number of TCP connections (*i.e.,* flows), thereby degrading the performance of other cloud applications. There have also been a number of works devoted to providing minimum bandwidth guarantees [7] for tenant virtual machines (VMs) and achieving the *VM-level* fairness [8] or *tenant-level* fairness [9]. Nevertheless, these solutions cannot provide the fairness of application performance (*e.g.,* completion time or throughput). Moreover, a number of existing bandwidth sharing solutions (*e.g.,* [8]) are based on the "ideal" *hose model*, where all VMs are connected to a non-blocking logical switch through dedicated network connections. As a result, there have been little attention paid to achieving the fairness of application performance and bandwidth allocation on *congested* switch links.

To solve the issues above, in this paper, we present a utility max-min fair bandwidth allocation algorithm for cloud applications in sharing datacenter network resources. Specifically, by presenting a rigorous definition of *application-level utility max-min fairness*, we develop a non-linear model to study the relationship between the fairness of application performance (utility) and bandwidth allocation of applications. Based on such analysis, we further develop a simple yet effective bandwidth allocation algorithm using Newton's method [10] and implement our algorithm in an OpenFlow [11] controller. Extensive experiment results demonstrate that our algorithm can reduce the variation of application performance (utility) by 5.8 % – 10.8 %, compared with the traditional TCP rate control mechanism and flow-level utility max-min fair allocation algorithm, thereby achieving utility max-min fairness of bandwidth allocation for applications in datacenter networks.

The rest of this paper is organized as follows. Section 2 presents a bandwidth allocation model to analyze the relationship between application bandwidth and the fairness of application utility (*i.e.,* performance), which enables the design of our application-level utility max-min fair bandwidth allocation algorithm. Section 3 evaluates the effectiveness and overhead of our algorithm. Section 4 discusses our contribution in the context of related work. Finally, we conclude this paper and discuss our future work in Sect. 5.

2 Achieving Application-Level Utility Max-Min Fairness of Bandwidth Allocation

In this section, we first present a model of *utility max-min fairness* of bandwidth allocation among cloud applications. Next, we devise a simple yet effective algorithm to achieve the application-level utility fairness of bandwidth allocation in datacenter networks.

2.1 Network Bandwidth Allocation Model for Applications

We consider the datacenter network with the representative tree topology, hosting a set of running applications, denoted by $\mathcal{N} = \{1, 2, \cdots, N\}$. Each application $i \in \mathcal{N}$ initiates a number of network flows, denoted by $\mathcal{F}_i = \{1, 2, \cdots, m_i\}$. Each flow is denoted by a two-tuple (i, f), representing that the flow is the f-ordered in $\mathcal{F}_i$. We use a binary variable $h^l_{i,f}$ to denote whether the flow (i, f) of an application i passes through the link l. We also use C_l and $\alpha_{i,f}$ to denote the bandwidth capacity of a network link l and the bandwidth allocated to a flow (i, f), respectively.

$$h^l_{i,f} = \begin{cases} 1, & \text{the flow } (i,f) \text{ passes through the link } l \\ 0, & \text{otherwise} \end{cases}$$

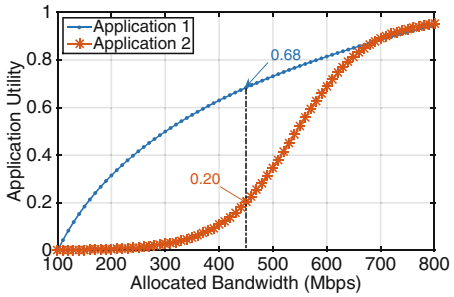

Fig. 1. Application utility: the application performance achieved by allocating different amount of network bandwidth.

Typically, the datacenter is hosting various types of workloads, ranging from CPU-intensive, data-intensive, to latency-sensitive applications. Different applications are able to achieve *different performance* when allocated the *same amount of network bandwidth*. As shown in Fig. 1, application 1 is able to achieve better application utility (*i.e.,* performance) than application 2, when allocated the same bandwidth of 450 Mbps. In this paper, we use *application utility* to measure the application performance according to the allocated network bandwidth. Specifically, with a particular focus on big data applications, we leverage the log function to formulate the application utility (*i.e., utility function* [12] $f_i(\cdot)$ of an application i) as below,

$$f_i(\alpha_i) = \log_{r_i} \alpha_i, \tag{1}$$

where α_i denotes the network bandwidth allocated to the application i, and r_i denotes the bandwidth demand of the application i, which is limited by the bandwidth capacity C_l and the aggregated bandwidth demand $r_{i,f}$ of application flows. Accordingly, we have $r_i = \min(\sum_f r_{i,f}, C_l)$. Similarly, we formulate the *flow utility* [13] (utility function $f_{i,f}(\cdot)$ of an application flow (i, f)) as $f_{i,f}(\alpha_{i,f}) = \log_{r_{i,f}} \alpha_{i,f}$, where $r_{i,f}$ is the bandwidth demand of the flow (i, f).

Table 1. Key notations in our bandwidth allocation model.

Notation	Definition
$\mathcal{N}$	Set of applications in the datacenter
$\mathcal{N}_l$	Application set running on a link l
$\mathcal{F}_i$	Network flow set of an application i
$\mathcal{L}$	Set of links that host application flows in the network
r_i	Bandwidth demand of an application i
$r_{i,f}$	Bandwidth demand of a flow f of an application i
α_i	Bandwidth allocated to an application i
$\alpha_{i,f}$	Bandwidth allocated to a flow f of an application i
t_i	Bandwidth temporarily allocated to an application i
$t_{i,f}$	Bandwidth temporarily allocated to a flow f of an application i
C_l	Bandwidth capacity of a link l
$h_{i,f}^l$	Whether the flow f of an application i passes through the link l
$f_i(\cdot)$	Bandwidth utility function of an application i

Using the notations of application utility defined in Eq. (1), we proceed to define the *application-level utility max-min fairness* of the bandwidth allocation for different cloud applications running on a *congested* network link.

Definition 1. Application-level Utility Max-min Fairness: Given a feasible application bandwidth allocation vector $\mathbf{a} = (\alpha_1, \alpha_2, \cdots, \alpha_n)$, the utility-ordered application bandwidth allocation vector is $\overline{\mathbf{a}} = (\alpha_{r_1}, \alpha_{r_2}, \cdots, \alpha_{r_n})$, such that $f_{r_k}(\alpha_{r_k}) \leq f_{r_{k+1}}(\alpha_{r_{k+1}}), \forall k \in [1, n-1]$. An application-level utility max-min fair allocation vector is the largest feasible utility-ordered vector in the network bandwidth allocation space.

In particular, given two feasible utility-ordered vectors $\overline{\mathbf{a}_i} = (\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_n})$ and $\overline{\mathbf{a}_j} = (\alpha_{j_1}, \alpha_{j_2}, \cdots, \alpha_{j_n})$, we say $\overline{\mathbf{a}_i} > \overline{\mathbf{a}_j}$ if and only if there exists m such that $f_{i_k}(\alpha_{i_k}) = f_{j_k}(\alpha_{j_k}), \forall k \in [1, m)$, and $f_{i_l}(\alpha_{i_l}) > f_{j_l}(\alpha_{j_l}), \forall l \in [m, n]$. The important notations used in our model are summarized in Table 1.

Based on Definition 1, in order to achieve the application-level utility max-min fairness of bandwidth allocation on a *congested* link l (*i.e.,* the sum of application bandwidth demand exceeds the network link capacity), we formulate the application bandwidth allocation problem as below,

$$\begin{cases} f_i(\alpha_i) - f_j(\alpha_j) = 0, & \forall i, j \in \mathcal{N}_l \\ \sum\limits_{i \in \mathcal{N}_l} \alpha_i - C_l = 0, \end{cases} \tag{2}$$

In addition, we use a vector $\mathbf{a}$ to denote the bandwidth allocation set (α_i, $\forall i \in \mathcal{N}_l$), and $\mathbf{G}(\cdot)$ to denote the left parts of the equation arrays formulated above. As a result, the model we formulated in Eq. (2) can be simplified as

$$\mathbf{G}(\mathbf{a}) = \mathbf{0} \tag{3}$$

Remark 1. The bandwidth allocation problem formulated above is a *non-linear* model, which is difficult and time-consuming to solve. As our objective is to allocate network bandwidth for applications without bringing much computational overhead, we seek to design a heuristic algorithm that can be implemented in a real-world datacenter. In particular, if a network link l is not a *congested* link (*i.e.,* the sum of application bandwidth demand is less than the bandwidth capacity of the link l), the demands of application flows passing through this link can be satisfied directly.

2.2 Bandwidth Allocation Algorithm for Achieving Application-Level Utility Max-Min Fairness

As we have analyzed in Sect. 2.1, the bandwidth allocation problem formulated in Eq. (2) is hard to solve in polynomial time. To allocate the network bandwidth for applications in practice, we design a simple yet effective algorithm in Algorithm 1 to achieve application-level utility max-min fairness based on Newton's method [10]. The detailed procedure of our algorithm is elaborated as follows.

Consider a set of applications $\mathcal{N}$ with the bandwidth demand of network flows $r_{i,f}$ running on the datacenter links $\mathcal{L}$. For each *congested* network link l in the datacenter, Algorithm 1 first initializes several algorithm parameters, such as the iterator k. Using Newton iteration method [10], Algorithm 1 then calculates the temporary bandwidth allocation $t_{i,f}$ of network flows running on the link l. Specifically, for each iteration k, the temporary bandwidth allocation vector $\mathbf{t}^{(k)} = (t_1, t_2, \cdots, t_{N_l})$ for applications running on a link l is obtained by

$$\mathbf{t}^{(k)} = \mathbf{t}^{(k-1)} - \mathbf{A}_k \mathbf{G}(\mathbf{t}^{(k-1)}), \tag{4}$$

where $\mathbf{A}_k$ is a matrix used in Newton's method [10] that avoids the inverse calculation of the matrix $\mathbf{G}(\mathbf{t})$, and $\mathbf{G}'(\mathbf{t})$ is the Jacobian matrix of $\mathbf{G}(\mathbf{t})$. They can be calculated by Eqs. (5) and (7), respectively.

$$\mathbf{A}_k = 2\mathbf{A}_{k-1} - \mathbf{A}_{k-1}\mathbf{G}'(\mathbf{t}^{(k)})\mathbf{A}_{k-1}, \tag{5}$$

$$\mathbf{A}_0 = \mathbf{G}'(\mathbf{t}^{(0)})^{-1}, \tag{6}$$

$$\mathbf{G}'(\mathbf{t}) = \begin{pmatrix} \frac{\partial f_1}{\partial t_1} & \frac{\partial f_2}{\partial t_2} & 0 & \cdots & 0 \\ \frac{\partial f_1}{\partial t_1} & 0 & \frac{\partial f_3}{\partial t_3} & \cdots & 0 \\ \vdots & & & & \\ \frac{\partial f_1}{\partial t_1} & 0 & 0 & \cdots & \frac{\partial f_{N_l}}{\partial t_{N_l}} \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}. \tag{7}$$

Algorithm 1. Achieving utility max-min fairness of bandwidth allocation among applications

Input: Application set $\mathcal{N}$, network link set $\mathcal{L}$, the relationship between flows and links $h_{i,f}^{l}$, the flow bandwidth demand $r_{i,f}$.
Output: Network bandwidth allocation for the application set $\mathcal{N}$.

while exists unallocated application flows ($\mathcal{N} \neq \varnothing$) **do**
  for all *congested* link $l \in \mathcal{L}$ **do**
    Initialize $k \leftarrow 0$, $\mathbf{t}^{(0)} \leftarrow$ the output of *binary search* algorithm, $\mathbf{A}_0 \leftarrow$ Eq. (6);
    Identify the application flows (i, f) on the link l (*i.e.,* $h_{i,f}^{l} \equiv 1$);
    while $k \leq T$ &&$\|\mathbf{t}^{(k)} - \mathbf{t}^{(k-1)}\|_1 > \delta$ **do**
      $\mathbf{t}^{(k)} \leftarrow$ Eq. (4); $\mathbf{A}_k \leftarrow$ Eq. (5); $k++$;
    end while
    $t_{i,f} \leftarrow$ Eq. (8);
  end for
  Identify the flow $(i, f)_{\min}$ with the minimum utility $f_{i,f}(t_{i,f})$;
  Allocate the minimum temporary bandwidth along the flow path $l \in \mathcal{L}$ to the flow $(i, f)_{\min}$, *i.e.,* $\alpha_{i,f} = \min(t_{i,f})$;
  if all the flows of an application i have been allocated bandwidth **then**
    Remove the application i from the application set $\mathcal{N}$;
  end if
end while
return the bandwidth $\alpha_{i,f}$ allocated to network flows of applications.

The iteration terminates in two conditions: (1) The iteration exceeds a maximum T. (2) The difference of iteration outputs $\mathbf{t}$ (*i.e.,* the first order norm of $\mathbf{t}^{(k)} - \mathbf{t}^{(k-1)}$) is less than a small value δ. In proportional to the bandwidth demand of network flows on the link l, the temporary bandwidth allocation $t_{i,f}$ is given by

$$t_{i,f} = t_i \cdot \frac{r_{i,f}}{\sum_{f \in \mathcal{F}_i} r_{i,f} h_{i,f}^{l}} \tag{8}$$

Finally, to achieve the application-level utility max-min fairness in Definition 1, Algorithm 1 identifies the flow $(i, f)_{\min}$ with the minimum flow utility and allocates the minimum temporary bandwidth $\min(t_{i,f})$ along the flow path to the flow $(i, f)_{\min}$. An application i requires to be removed from the set $\mathcal{N}$, if all its application flows have been allocated bandwidth. In particular, Algorithm 1 iteratively executes the procedure above until all the application flows are allocated network bandwidth.

Remark 2. First, Algorithm 1 can be periodically executed in the case that (1) new applications are running in the datacenter, and (2) the bandwidth demand of application flows has been changed. The algorithm execution period can be adjusted by the datacenter operator, ranging from several minutes to one hour. *Second*, given the detailed notations in Table 1, the complexity of Algorithm 1 is in the order of $\mathcal{O}(|\mathcal{N}| \cdot |\mathcal{L}|)$. In practice, the *congested* links are mainly restricted to the top-layered links in the tree topology [14], and accordingly, Algorithm 1

only requires running on the top-layered links to obtain the bandwidth allocation of application flows. *Third*, Algorithm 1 can be implemented in an OpenFlow controller, such as Ryu[1] and OpenDaylight[2]. After calculating the bandwidth allocation for application flows by Algorithm 1, the application bandwidth can be limited by setting the `meter table` in the OpenFlow v1.3 switches.

3 Experimental Evaluation

In this section, we evaluate the effectiveness and computational overhead of our bandwidth allocation algorithm in the context of application utility, algorithm running time and underutilized network bandwidth. We compare the evaluation results of our algorithm with that of the conventional TCP rate control mechanism and flow-level utility max-min fair allocation in large-scale simulations.

Experimental Setup. We set up a datacenter network with the representative tree topology in Mininet v2.1.0+[3] virtual network environment, by varying the network scale from 100 switches to 800 switches. We use the iPerf application[4] to generate the different network flows for 10 running applications. Each flow is randomly set in the range of [1, 100] Mbps. We implement our algorithm with Ryu v3.9 OpenFlow controller and OpenFlow 1.3 Software Switch[5].

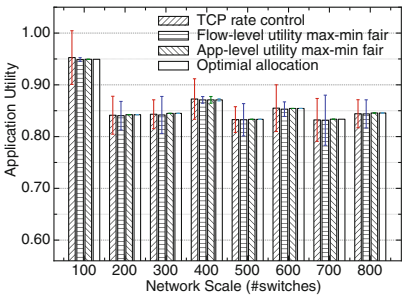

Fig. 2. Application utility achieved by different bandwidth allocation algorithms.

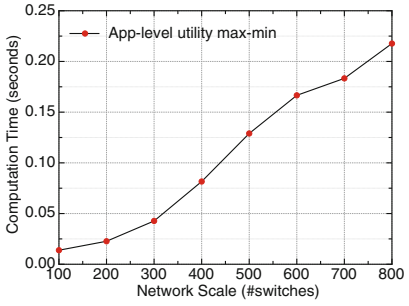

Fig. 3. Computation time of our bandwidth allocation algorithm.

Effectiveness and Computational Overhead. To examine the effectiveness of our bandwidth allocation algorithm, Fig. 2 compares the application utility achieved by three different algorithms. We observe that our application-level utility fair allocation algorithm significantly reduces the variation (*i.e.,* unfairness)

[1] http://osrg.github.io/ryu/.
[2] https://www.opendaylight.org.
[3] http://mininet.org.
[4] https://iperf.fr.
[5] http://cpqd.github.io/ofsoftswitch13/.

of application utility by 5.8 % – 10.8 %, compared with TCP rate control and flow-level utility max-min fair allocation algorithms, though the average application utility is comparable among different algorithms for each network scale. The rationale is that, our algorithm allocates the network bandwidth based on the aggregated flows of each application, and achieves utility max-min fairness for applications on congested links to circumvent the unfairness of application performance. Moreover, our algorithm achieves the application utility close to the optimal allocation, yet with a rough linear computational overhead of the number of network switches, as shown in Fig. 3. Such an overhead is consistent with the complexity analysis of our algorithm in Sect. 2.2. Specifically, the complexity of Algorithm 1 is reduced to $\mathcal{O}(|\mathcal{L}|)$ as the number of applications is a constant ($|\mathcal{N}| = 10$) in our experiment. In particular, the algorithm running time is within 0.25 s as the network scale increases to 800 switches.

We next examine the underutilized network bandwidth achieved by our bandwidth allocation algorithm. As shown in Table 2, the link bandwidth of datacenter networks is not efficiently utilized, *i.e.,* the total underutilized bandwidth of our algorithm is ranging from 132 to 938 Gbps as the number of network switches increases from 100 to 800. As a result, our algorithm achieves the application-level utility max-min fairness of bandwidth allocation *at the cost of* underutilizing the network resources. We illustrate this tradeoff by considering an example shown in Fig. 4, in which application A has two flows from A1 to A3 and A2 to A3, and application B has two flows from B1 to B3 and B2 to B3. Each flow has the same bandwidth demand 100 Mbps. All the links have the same bandwidth capacity 100 Mbps. Assume two applications have the same utility function. Our algorithm allocates 25 Mbps to the flows (A1, A3) and (A2, A3) and allocates 50 Mbps to the flow (B1, B3) on the congested link (S1, S2). The flow (B2, B3) is allocated 50 Mbps on the link (S1, S5). Hence, the underutilized bandwidth of network links between switches is summed up to 100 Mbps with our algorithm, which accounts for 25 % of switch bandwidth resources in the datacenter.

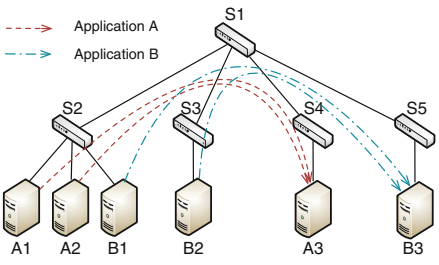

Fig. 4. Bandwidth allocation of two applications in an example tree topology.

Table 2. Underutilized network bandwidth achieved by our bandwidth allocation algorithm.

Scale (#switches)	Underutilized bandwidth
100	132 Gbps
200	214 Gpbs
300	331 Gbps
400	448 Gbps
500	692 Gbps
600	782 Gbps
700	808 Gbps
800	938 Gbps

4 Related Work

There have been a number of works on flow bandwidth allocation in traditional networks, with the aim of achieving max-min fairness [5], utility max-min fairness [13] and proportional fairness [6]. Several works have paid much attention to the *flow-level* utility. For example, Wang *et al.* [15] proposed an algorithm for guaranteeing the Quality of Service (QoS) of applications based on the flow-level utility. However, these works only provide the *flow-level* QoS and fairness, which is likely to impact the performance and fairness of applications with a number of network flows. Furthermore, the utility function of big data applications is quite different from that of the applications running in traditional networks.

With the evolution of datacenters, virtualization technique has been widely deployed to multiplex resources (*e.g.,* computing and bandwidth resources) and improve the utilization of servers. Recently, there have emerged a number of works that focus on providing bandwidth guarantees for tenant VMs in public clouds, such as deterministic guarantees [16], minimum guarantees [7], and proportional bandwidth share [9]. For example, Lam *et al.* [9] focused on achieving the tenant-level fairness of bandwidth allocation, while Shieh *et al.* [17] proposed a fair bandwidth share on the source VMs. Popa *et al.* [8] allocated bandwidth to achieve VM-pair level fairness. To cope with highly dynamic network traffic in datacenters, Guo *et al.* [18] proposed an efficient rate allocation algorithm to achieve both minimum guarantees and VM-pair level fairness. Different from these works, our work aims to achieve the *utility* max-min fairness among applications, in order to provide *performance guarantees* for cloud applications, while prior works provide bandwidth guarantees and achieve fairness at the tenant level or VM level. Moreover, prior works mainly implement the rate limiter on the Hypervisor [18], while our work makes an attempt to limit the bandwidth of network flows on the OpenFlow switches.

To provide application-level bandwidth guarantees, Kumar *et al.* [19] proposed to allocate bandwidth based on the communication patterns of big data applications, while Lee *et al.* [20] designed a VM placement algorithm to satisfy the bandwidth requirements of workloads. Chen *et al.* [21] presented a definition of performance-centric fairness and designed a bandwidth allocation strategy to achieve such fairness among applications. Different from prior works, our work defines the *application utility* to reflect the performance of various applications and achieves *utility max-min fairness* of bandwidth allocation, while [21] uses the reciprocal of the data transfer time to represent the application performance.

5 Conclusion and Future Work

To achieve the fairness of application performance, this paper proposes a rigorous definition of application-level utility max-min fairness, and design the utility fair bandwidth allocation algorithm that can be practically implemented in a real-world OpenFlow controller. Extensive experiment results using OpenFlow show that our bandwidth allocation algorithm can reduce the variation of application

utility by 5.8 % – 10.8 % and achieve utility max-min fairness of bandwidth allocation for applications, in comparison to the conventional TCP rate control mechanism and flow-level utility max-min fair allocation algorithm.

As our future work, we plan to investigate the tradeoff between high utilization and utility max-min fairness of bandwidth allocation for applications. We also plan to deploy our bandwidth allocation algorithm in real OpenFlow switches and evaluate its effectiveness and running overhead.

Acknowledgments. The research was supported by a grant from the National Natural Science Foundation of China (NSFC) under grant No.61502172, and by a grant from the Science and Technology Commission of Shanghai Municipality under grant No.14DZ2260800. The corresponding author is Fei Xu.

References

1. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. ACM Commun. **51**(1), 107–113 (2008)
2. Yi, X., Liu, F., Liu, J., Jin, H.: Building a network highway for big data: architecture and challenges. IEEE Netw. Mag. **28**(4), 5–13 (2014)
3. Roy, A., Zeng, H., Bagga, J., Porter, G., Snoeren, A.C.: Inside the social network's (datacenter) network. In: Proceedings of SIGCOMM, pp. 123–137, August 2015
4. Xu, F., Liu, F., Jin, H., Vasilakos, A.V.: Managing performance overhead of virtual machines in cloud computing: a survey, state of art and future directions. Proc. IEEE **102**(1), 11–31 (2014)
5. Bertsekas, D.P., Gallager, R.G.: Data Network, 2nd edn. Prentice-Hall, London (1992)
6. Kelly, F.P., Maulloo, A.K., Tan, D.K.H.: Rate control for communication networks: shadow price, proportional fairness and stability. J. Oper. Res. Soc. **49**(3), 237–252 (1998)
7. Guo, J., Liu, F., Tang, H., Lian, Y., Jin, H., Lui, J.: Falloc: fair network bandwidth allocation in iaas datacenters via a bargaining game approach. In: Proceedings of ICNP, pp. 1–10, October 2013
8. Popa, L., Kumar, G., Chowdhury, M., Krishnamurthy, A., Ratnasamy, S., Stoica, I.: FairCloud: sharing the network in cloud computing. In: Proceedings of SIGCOMM, pp. 187–198, August 2012
9. Lam, T., Radhakrishnan, S., Vahdat, A., Varghese, G.: Netshare: Virtualizing Data Center Networks across Services. Technical Report CS2010-0957, Department of Computer Science and Engineering, University of California, San Diego (2010)
10. Wang, X.H., Han, D.F., Sun, F.Y.: Point estimates on deformation newton's iterations. Mathematica Numerica Sinica **1**(2), 145–156 (1990)
11. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
12. Shenker, S.: Fundamental design issues for the future internet. IEEE J. Sel. Areas Commun. **13**(7), 1176–1187 (1995)
13. Cao, Z., Zegura, E.: Utility max-min: an application-oriented allocation scheme. In: Proceedings of Infocom, pp. 793–801, April 1999

14. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. ACM SIGCOMM Comput. Commun. Rev. **38**(4), 63–74 (2008)
15. Wang, W., Palaniswami, M., Low, S.: Application-oriented flow control fundamentals algorithms and fairness. IEEE/ACM Trans. Netw. **14**(6), 1282–1291 (2006)
16. Ballani, H., Costa, P., Karagiannis, T., Rowstron, A.: Towards predictable datacenter networks. ACM SIGCOMM Comput. Commun. Rev. **41**(4), 242–253 (2011)
17. Shieh, A., Kandula, S., Greenberg, A., Kim, C., Saha, B.: Sharing the data center network. In: Proceedings of NSDI, pp. 309–322, March 2011
18. Guo, J., Liu, F., Huang, X., Lui, J., Hu, M., Gao, Q., Jin, H.: On efficient bandwidth allocation for traffic variability in datacenters. In: Proceedings of Infocom, pp. 1572–1580, April 2014
19. Kumar, G., Chowdhury, M., Ratnasamy, S., Stoica, I.: A case for performance-centric network allocation. In: Proceedings of HotCloud, pp. 9–9, June 2012
20. Lee, J., Turner, Y., Lee, M., Popa, L., Banerjee, S., Kang, J.M., Sharma, P.: Application-driven bandwidth guarantees in datacenters. In: Proceedings of SIGCOMM, pp. 467–478, August 2014
21. Chen, L., Feng, Y., Li, B., Li, B.: Towards performance-centric fairness in datacenter networks. In: Proceedings of Infocom, pp. 1599–1607, April 2014