Fair Network Bandwidth Allocation in IaaS Datacenters via a Cooperative Game Approach

Jian Guo, Fangming Liu*, *Member, IEEE,*
John C.S. Lui, *Fellow, IEEE, ACM,* and Hai Jin, *Senior Member, IEEE*

***Abstract*—With wide application of virtualization technology, tenants are able to access isolated cloud services by renting the shared resources in IaaS datacenters. Unlike resources such as CPU and memory, datacenter network, which relies on traditional transport-layer protocols, suffers unfairness due to a lack of VM-level bandwidth guarantees. In this paper, we model the datacenter bandwidth allocation as a cooperative game, towards VM-based fairness across the datacenter with two main objectives: (i) guarantee bandwidth for VMs based on their base bandwidth requirements, and (ii) share residual bandwidth in proportion to the weights of VMs. Through a bargaining game approach, we propose a bandwidth allocation algorithm, *Falloc*, to achieve the asymmetric Nash bargaining solution (NBS) in datacenter networks, which exactly meets our objectives. The cooperative structure of algorithm is exploited to develop an online algorithm for practical real-world implementation. We validate *Falloc* with experiments under diverse scenarios, and show that by adapting to different network requirements of VMs, *Falloc* can achieve fairness among VMs and balance the tradeoff between bandwidth guarantee and proportional bandwidth sharing. Our large scale trace-driven simulations verify that *Falloc* achieves high utilization while maintaining fairness among VMs in datacenters.**

***Index Terms*—IaaS datacenter, fairness, bandwidth allocation, Nash bargaining solution.**

I. Introduction

Infrastructure-as-a-service (IaaS) cloud services, such as Amazon EC2 [1], have become an attractive choice for today's business, in which large-scale datacenters are multiplexing computing, storage and network resources across multiple tenants. With a simple pay-as-you-go charging model, tenants are able to rent their respective sets of virtual machines (VMs) with performance isolation on CPU and memory resources [2]. However, in current datacenters, the scarce network bandwidth is shared across many tenants without any performance guarantees [2], [3]. The bandwidth between VMs can fluctuate significantly due to the competition of network intensive applications, and their performance may become unpredictable. The uncertainty in the execution times of jobs increases the risk of revenue loss for tenants, which is against the provider's incentive to attract more tenants. As the rapid development in IaaS cloud, it is critical to reason about how to properly share networks in IaaS datacenters.

Towards providing predictable performance *for the tenants* under multiplexed infrastructure, we need to take *fairness* into consideration due to the limited bandwidth resources in datacenters. The essential of fairness is to guarantee each application's performance in the competition of various network aggressive applications. Current cloud applications have two primary requirements for fairness [4], which can be used to form the basis of designing an allocation policy: *minimum bandwidth guarantee* and *proportional bandwidth share*.

Minimum bandwidth guarantee can provide strong isolation among VMs or tenants, since it ensures a lower bound of bandwidth allocation independent of the communication patterns of other VMs. With the ability of guaranteeing bandwidth for each VM, providers can negotiate SLAs on network performance with the tenants, which may be attractive to tenants deploying network intensive applications in the cloud. Proportional share, on the other hand, is to share bandwidth in proportion to certain associated weights among VMs, where each VM can obtain a portion of the physical bandwidth regardless of the competition at the flow-level. By slicing the network bandwidth, proportional share makes efficient use of the network resource and maintains weighted fairness among the VMs, which can be useful to differentiate the service level for applications with different priorities.

For cloud providers, we need to achieve high network utilization throughout the datacenter. The network resources should be fully allocated among VMs if there exist unsatisfied demands. If a VM is able to utilize the residual bandwidth left by other idle VMs, the increase in throughput will shorten the completion times of jobs that are bottlenecked by network resources. In this way, more applications or VMs can be deployed on the same infrastructure, which will further increase the providers' revenues.

In existing networks, such as Internet, the TCP-friendly sharing naturally provides applications with max-min fairness, which meets the requirements for flow-level fairness and work-conserving sharing. However, in IaaS datacenters which require fairness among VMs, the tenants may suffer unpredictable performance, as the bandwidth is shared in proportion to the number of TCP flows in VMs without guarantee. Due to the lack of VM-level rate-control protocols, the bandwidth allocation in IaaS datacenters raises great challenges in simultaneously guaranteeing VM network performance and achieving high network utilization under unpredictable traffic demand. For example, existing reservation policy (e.g., [5])

J. Guo, F. Liu, and H. Jin are with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China. E-mail: {guojian, fmliu, hjin}@hust.edu.cn.

J. C.S. Lui is with the Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N.T, Hong Kong. E-mail: cslui@cuhk.edu.hk.

leads to bandwidth wastage if the reserved bandwidth is not fully utilized.

Moreover, there is a tradeoff between guaranteeing bandwidth and sharing bandwidth proportionally [4]. For instance, proportional sharing (e.g., [6]) ensures a certain portion of the shared bandwidth for each VM. However, the VM can hardly get a minimum bandwidth, because the shared portion reduces as more VMs are competing for the same physical link. Consequently, the bandwidth allocation should offer mechanisms to balance such a tradeoff while achieving the fairness requirements.

While existing works focus on providing bandwidth isolation technologies for VMs, they fail to take a theoretical insight into such a fair resource sharing problem. In this paper, we view the bandwidth allocation as a basic resource sharing problem involving consideration of fairness and utilization.

By taking advantage of a game-theoretical approach, this paper takes the first step to model the bandwidth allocation process in datacenters as a *Nash bargaining game*, where all VMs are cooperative so as to maximize the social welfare, i.e., the network utilization, with fairness among VMs being guaranteed. In summary, we make the following contributions in this work:

- We apply rigorous game-theoretic techniques to model and solve datacenter network resource sharing problem by considering both efficiency and fairness. For the key network requirements of applications in the cloud, bandwidth guarantee and proportional bandwidth sharing, we consider them in a cooperative game-theoretic framework by defining the base bandwidth and the weight for each VM, and show how to achieve the Nash bargaining solution for bandwidth sharing.
- Based on a bargaining game approach, we present the design of *Falloc*, an application layer bandwidth allocation algorithm to achieve fairness among VMs in datacenters, which corresponds to the weighted Nash bargaining solution. Our offline algorithm of *Falloc* can guarantee the bandwidth of a VM when its bandwidth requirement is less than the base bandwidth, and share the residual bandwidth among VMs in proportion to their weights.
- To realize *Falloc* in datacenter networks, we develop a distributed online algorithm using the cooperative structure of the offline algorithm. The online algorithm can theoretically achieve the same solution as the offline algorithm, and we experimentally show that the online algorithm can control the relative error of rate within 6% as compared to the offline algorithm.
- We implement the *Falloc* prototype and evaluate it on a testbed under diverse scenarios. By characterizing the impact of both the base bandwidth and the weight on bandwidth allocation, we validate *Falloc*'s ability to enforce bandwidth guarantee and proportional network sharing, and show that *Falloc* can balance the tradeoff by adjusting the base bandwidth of VMs. We carry out trace-driven simulations using Mapreduce workloads and show that *Falloc* can adapt to dynamic traffic. It can achieve a utilization approximate to the best effort manner while providing performance guarantees for VMs by enforcing a fair bandwidth allocation.

The rest of this paper is organized as follows. In Sec. II, we present the motivation and the objectives of our work. In Sec. III, we formulate the datacenter network model and the optimization for fair bandwidth allocation. Sec. IV presents the solution to the optimization problem via a bargaining game approach. Based on an offline algorithm for the optimal solution, we complement our preliminary work [7] and develop a practical online algorithm in Sec. V. We evaluate our proposed algorithms in Sec. VI with comprehensive experiments and simulations. Related work is presented in Sec. VII, and Sec. VIII concludes.

II. Motivation and Design Objectives

A. Motivation and Objectives: Fair Datacenter Networks

As applications are run by VMs in IaaS datacenters, providers can reserve certain bandwidth for different VMs, or allocate a certain portion of the bandwidth on congested links to VMs based on the bandwidth requirements of these applications. However, it is not easy to achieve fairness relying on traditional Transmission Control Protocol (TCP). The challenge comes from the fact that the TCP protocol maintains flow-level fairness and one cannot change this protocol if he wants to run any TCP-based applications in the cloud.

Hence, to fairly share the intra-datacenter networks, we need to design a *VM-level bandwidth sharing policy* with the ability to fulfill the bandwidth requirements of VMs running different applications. The following important objectives should be taken into consideration:

- **Bandwidth guarantee.** With bandwidth guarantee for VMs, tenants can achieve predictable performance for network sensitive applications running in these VMs. For example, a web service can provide fast and stable data delivery to users if the data transfer between the server's front and back end is guaranteed. By deploying bandwidth guarantee for the VM instance, cloud providers are able to provide quantitative network performance for cloud applications, thus to attract more tenants.
- **Weight assignment.** Since jobs in the cloud have different priorities, the policy should have the ability to assign differentiate weights to VMs running different applications. For example, an important job calculating stock prices and a less urgent data-backup job are sharing the same congested link, the cloud provider may want to allocate more bandwidth for the important job. With the weights of VMs, the policy should share the bandwidth in proportion to their respective weights.

Considering the tradeoff between these two objectives, an alternative approach is to combine them together and use each on a ratio of the total bandwidth according to the applications' network requirements. Since proportional bandwidth share can highly utilize the bandwidth on congested link, we can prioritize bandwidth guarantee for VMs, and share the residual bandwidth in proportion to the weight of each VM. This way, we can achieve a high utilization of the network bandwidth as well as maintain fairness among VMs.

B. Design Choices: Base Bandwidth and Weight for VMs

We now introduce the definition of *base bandwidth* (B) and *weight* (K). The base bandwidth of each VM is a threshold of guaranteeing bandwidth for a VM. The weight of each VM represents the portion of shared bandwidth for a VM. Specifically, if the bandwidth demand of a VM is lower than the base bandwidth, we allocate sufficient bandwidth to the VM to satisfy its network requirement. Otherwise, we allocate bandwidth higher than the base bandwidth to these VMs, and the part beyond the base bandwidth is shared in proportion to the weight by slicing the residual bandwidth left by allocating the base bandwidth. The motivation of introducing B and K into the datacenter network model can be summarized as below:

Towards providing SLA. Today's IaaS cloud platforms do not provide SLAs on network bandwidth. By providing a guaranteed bandwidth B and a weight K for VMs, cloud providers are able to price the bandwidth and cloud users can choose suitable bandwidth for VMs according to the requirements of network performance of their applications. This way, we provide incentives for users to migrate their business to the cloud, which can potentially increase the provider's revenue.

Working with the VM allocation based methods. The base bandwidth can bridge the VM allocation based mechanisms (e.g., [5], [8]) with our competition based mechanisms. A VM allocation based mechanism studies how to place VMs to the servers, and reserve bandwidth for these VMs. The reserved bandwidth for each VM needs to be specified in advance. After the placement, the reserved bandwidth can be treated as the base bandwidth in our model. However, instead of enforcing a bandwidth reservation for each VM statically, we slice the physical bandwidth dynamically to the VMs according to their bandwidth demand. For example, Oktpus [5] proposes a virtual cluster in which each VM is connected to a virtual switch with bandwidth $\tilde{B}$. We can set $B = \tilde{B}$ as the base bandwidth and then apply our policy to allocate the unused bandwidth to the VMs with bandwidth demand higher than the base bandwidth.

Flexible fairness. Due to the limited network resources in datacenters, it may not be possible to guarantee the whole of the bandwidth demand for each VM. A practical strategy is to guarantee a certain bandwidth considering both the tenant's budget as well as the bandwidth demand of the application, and use another economical policy for bandwidth demand beyond the guaranteed bandwidth. Hence, our choice is to guarantee a certain bandwidth within the base bandwidth demand for each VM and share the residual bandwidth in proportion to VMs' weights.

III. Bandwidth Sharing in Datacenter Network

We first take a rigorous look into the underlying resource sharing problem in the context of IaaS datacenters.

A. Datacenter Network Model

Hose model for DCN. Existing works such as multi-path routing (e.g., [9]) and multi-tree topologies (e.g., [10]) have demonstrated the feasibility of building high bisection bandwidth of datacenter networks, where congestions seldom happen inside the networks fabric. Hence, we design the bandwidth allocation algorithm based on a hose model abstraction of datacenter networks as shown in Fig. 1 (which is also used by recent proposals [5], [11]). In the hose model, the servers are connected to a non-blocking virtual switch and the physical bandwidth of servers can be fully utilized without considering the bottlenecks inside the datacenter networks.

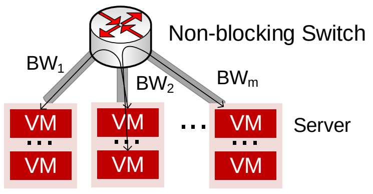


Fig. 1. Datacenter model with m servers connecting to a non-blocking switch with heterogeneous bandwidth. The VMs are hosted on these servers.

Given that the average utilization of 80% links at core/aggregation layers is less than 10%, and 75% traffic is within the same rack [12], the hose model is reasonable for datacenter traffic. In consequence, we can focus on bandwidth allocation for VMs on end servers, and leave the routing of packets to existing routing or load balancing algorithms inside the networks (e.g., [9], [10]).

Pairwise bandwidth allocation. In our model, we choose to allocate bandwidth for each VM-pair (e.g., [13]), which provides more fine-grained guarantees compared to the solution of allocating bandwidth to the aggregated traffic of each VM (e.g., [5]). For example, a reduce task of a MapReduce job on VM X needs to shuffle data from multiple map tasks on different VMs. If we only guarantee the aggregated traffic of X, and the bandwidth between X and each VM with map tasks is not guaranteed, then the job may be delayed by one or more slow tasks on network congested VMs. Hence, the pairwise bandwidth allocation can guarantee the performance of applications when a VM communicates with multiple VMs.

We consider an IaaS cloud model consisting of M servers $\mathcal{M} = \{1, 2, \ldots, M\}$ and N VMs $\mathcal{N} = \{1, 2, \ldots, N\}$ hosted by these servers. Let $D_N = [D_{i,j}]_{N \times N}$ be a matrix representing the bandwidth demand between VMs in a datacenter, where $D_{i,j}$ is the bandwidth demand of VM-pair from VM i to VM j. We specify a bandwidth allocation strategy by solving a rate matrix $r_N = [r_{i,j}]_{N \times N}$, where $r_{i,j}$ is the bandwidth allocation from VM i to j. To distinguish the ingress and egress bandwidth (or rate) of VMs (or servers), we use the superscripts I and E in the following problem formulation, respectively.

We summarize commonly used notations in Table I. VM i is denoted by a 7-tuple, $(r_i^I, r_i^E, D_i^I, D_i^E, B_i^I, B_i^E, K_i)$, and server m is denoted by a 3-tuple (C_m^I, C_m^E, V_m). For the total ingress and egress bandwidth demand of VM i, we have $D_i^I = \sum_{k=1}^N D_{k,i}$ and $D_i^E = \sum_{k=1}^N D_{i,k}$. Similarly, the total ingress and egress rates of VM i are $r_i^I = \sum_{k=1}^N r_{k,i}$ and $r_i^E = \sum_{k=1}^N r_{i,k}$, respectively. We use $V_m \subset \mathcal{N}$ to denote the set of VMs on server m. Suppose each server is equipped with a

TABLE I
NOTATIONS

Notations	Definitions
$D_{i,j}$	Bandwidth demand from VM i to j
$r_{i,j}$	Bandwidth allocation (rate) from VM i to j
$D_i^E(D_i^I)$	Egress (ingress) bandwidth demand of VM i
$r_i^E(r_i^I)$	Egress (ingress) bandwidth allocation of VM i
$B_i^E(B_i^I)$	Egress (ingress) base bandwidth of VM i
K_i	Weight of VM i
$C_m^E(C_m^I)$	Egress (ingress) bandwidth capacity of server m
V_m	The set of VMs on server m

full-duplex Ethernet adapter, then we have $C_m = C_m^I = C_m^E$.

As the bandwidth guarantee acts as an SLA between the cloud provider and tenants, the provider should guarantee the base bandwidth even in extreme cases that all the traffic demands of VMs are aggressive. Hence, when allocating the VMs to physical servers, the provider should maintain the sum of base bandwidth of all the VMs hosted on the same server less than the physical bandwidth of this server, i.e., $\sum_{i \in V_m} B_i < C_m$.

In the formulation part, we focus on the bandwidth allocation with instantaneous bandwidth demand at a specific time. We will provide strategies to implement our protocol under dynamic bandwidth demand in the algorithm design, and evaluate the performance in the simulations.

B. Partition of Bandwidth Requirement

As presented in Sec. III-A, the base bandwidth B_i of each VM i and the bandwidth demand $D_{i,j}$ of each VM-pair from VM i to j are given. The allocation process can be viewed as a bandwidth competition among $N \times N$ VM-pairs.

Let $B_{i,j}$ represent the base bandwidth for the VM-pair from VM i to VM j, which is unknown and can be derived from the base bandwidth of VM i and VM j. In our work, we specify $B_{i,j}$ as the following form

$$B_{i,j} = \min\{B_i^E \frac{K_i}{\sum_{D_{ik} \neq 0, k \in \mathcal{N}} K_k}, B_j^I \frac{K_j}{\sum_{D_{kj} \neq 0, k \in \mathcal{N}} K_k}\}, \tag{1}$$

where $D_{i,j} \neq 0$ implies that VM i has connections to VM j. Similarly, we have the weight for the VM-pair from VM i to VM j

$$K_{i,j} = \frac{K_i}{\sum_{D_{ik} \neq 0, k \in \mathcal{N}} 1} + \frac{K_j}{\sum_{D_{kj} \neq 0, k \in \mathcal{N}} 1}. \tag{2}$$

Note that $B_{i,j}$ and $K_{i,j}$ are only available when $D_{i,j} \neq 0$. The derivation is based on an idea of weight-based partition.

Since each VM may be in communication with one or more VMs, the base bandwidth $B_{i,j}$ of a VM pair can be viewed as (i) a portion of the egress base bandwidth of VM i, or (ii) a portion of the ingress base bandwidth of VM j. As we defined in Sec. (III-A), the base bandwidth and the weight represent SLAs on the bandwidth allocation between tenants and cloud providers. It indicates that a VM-pair could obtain a higher base bandwidth if it consists of VMs with higher weights. Hence, the portion of $B_{i,j}$ to B_i^E is set as the portion of the weight of VM j to all the destination VMs from source VM i, and the portion of $B_{i,j}$ to B_i^I is similar. Therefore, we can choose the smaller value to be $B_{i,j}$ for the case that the total base bandwidth exceeds the physical bandwidth, and causes failure to guaranteeing the base bandwidth $B_{i,j}$.

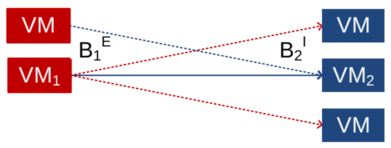


Fig. 2. An example of deriving $B_{i,j}$ and $K_{i,j}$ of VM-pair using the base bandwidth and the weight of VMs: if $B_1^E = B_2^I = 100$, $K_1 = K_2 = 1$, then we have $B_{1,2} = \min\{\frac{100}{3}, 50\}$ Mbps and $K_{i,j} = \frac{1}{3} + \frac{1}{2}$.

The derivation of the weights for VM-pairs in Eq. (2) is similar to the argument we provide for the base bandwidth. The weight of a VM is divided equally to all of its associated VM-pairs. As the weight of a VM-pair has no upper bound and should reflect the weights of both VMs, we choose the sum of the weight of both source VM and destination VM. This way, the weight of a VM group (rented by a tenant) will be equal to the total weight of the VMs in this group, and the shared bandwidth of this group will be proportional to the total weight of its VMs [4].

Fig. 2 illustrates an example of calculating $B_{1,2}$. All the egress and ingress base bandwidth of the VMs are 100 Mbps. We get $\frac{100}{3}$ Mbps by partitioning B_1^E and 50 Mbps by partitioning B_2^I, hence $B_{i,j}$ is $\frac{100}{3}$ Mbps. Similarly, all the weights of VMs are 1, then we have $K_{i,j} = \frac{1}{3} + \frac{1}{2}$.

C. Asymmetric NBS for Fairness

With consideration on both utilization and fairness, we choose to use the Nash bargaining solution in game theory to solve this bandwidth allocation problem. In the Nash bargaining game, two or more players enter the game with an initial utility and a utility function. They cooperate in the game to achieve a win-win solution, in which the social utility gains represented by the Nash product are maximized. This is exactly the situation of bandwidth allocation in datacenters, where each VM-pair should be guaranteed with an initial bandwidth, and the provider aims to maximize the joint profits associated with all the VM-pairs. Since NBS ensures the *Pareto optimality* and achieves the *fairness* in resource allocation, we believe that NBS is a suitable alternative for our allocation policy in the context of datacenter networks.

The Nash bargaining game was first presented in the context of communication networks by [14]. As shown in [15], [16], the solutions in game theory ensure Pareto optimality and fairness, which is equivalent to proportional fairness if the utility functions are logarithmic forms [17].

Based on the optimization problem in previous game-theoretical framework, we first present the main concepts from NBS. The N^2 VM-pairs can be viewed as the players who are competing for limited bandwidth resources in datacenters. Note that the bandwidth is the only performance metric in the model, we can use $r_{i,j}$ as a simple utility for each VM-pair. Since all the players have their respective weights, we apply the asymmetric weighted Nash bargaining solution [18] and assign them with different contributions to the social welfare by using the exponentiation of the utility gains, i.e., $(r_{i,j} - L_{i,j})^{K_{i,j}}$.

A major difference of the bandwidth allocation between datacenters and traditional networks is the rate control mechanism. Specifically, datacenters leverage rate limit on VM-to-VM traffic without changing rate control at transport layer. Hence, the bandwidth allocation should be aware of the traffic demand of each VM-pair, in case that the allocated bandwidth is not fully utilized. Here, we first assume that the traffic demand for each VM-pair is given, and apply the game-theoretical framework to achieving a solution which satisfies the bandwidth requirements of tenants. Then, we will develop an online algorithm which requires no demand prediction in Sec. V.

Considering such VMs whose bandwidth demands are less than the base bandwidths, the allocation should ensure an initial lower bound for each VM-pair, i.e., $r_{i,j} \geq L_{i,j}$ where

$$L_{i,j} = \min\{D_{i,j}, B_{i,j}\}. \tag{3}$$

The lower bound assures that the minimum allocation will not exceed the traffic demand, thus can be fully utilized.

As the bandwidth demand is the maximal rate that a VM-pair can achieve, the allocated rate $r_{i,j}$ should be upper bounded by $U_{i,j}$ where

$$U_{i,j} = \min\{D_{i,j}, C_m, C_l\},\ i \in V_m, j \in V_l. \tag{4}$$

After obtaining the domain of the rate for each VM-pair, i.e., $r_{i,j} \in [L_{i,j}, U_{i,j}]$, we use $X \subseteq \mathcal{R}^{N^2}$ to represent the vector space of the available allocation for N^2 VM-pairs, and then $r = \{r_{1,1}, r_{1,2}, \ldots, r_{n,n}\} \in X$ represents a specific allocation result.

Note that for a VM-pair whose bandwidth demand is lower than the base bandwidth, the lower bound and the upper bound are equivalent and the rate will be allocated equally to the bandwidth demand, i.e., $r_{i,j} = D_{i,j}$. However, for a VM-pair whose bandwidth demand is higher than the base bandwidth, the allocation policy should not only allocate the base bandwidth, but also try to allocate the exceeded bandwidth (i.e., $r_{i,j} - L_{i,j}$) in proportion to the weight $K_{i,j}$. From another perspective, the bandwidth sharing policy should maintain a high utilization in datacenter networks. Specifically, the allocation should be *Pareto-optimal*, where there exists no other allocation that leads to higher bandwidth for a VM-pair without sacrificing the bandwidth of other VM-pairs.

Accordingly, the corresponding initial utility for the VM-pair should be $L_{i,j}$. Let $x_0 = \{L_{0,0}, L_{0,1}, \ldots, L_{N,N}\} \in X$ be the vector of the initial utilities of all the VM-pairs. As each $r_{i,j}$ has a closed domain, the allocation space X is a convex and closed set. Let the set $G = \{r \mid r \in X, r_{i,j} \geq L_{i,j}, \forall i,j \in \mathcal{N}\}$ be the allocation results that each VM-pair can get at least their initial bandwidth. Suppose G is nonempty and then (G, x_0) is a bargaining game.

Definition 1. A function $\phi : (G, x_0) \rightarrow \mathcal{R}^N$ is called a Nash bargaining solution if it satisfies: $\phi(G, x_0) \in G$, Pareto optimality, symmetry, scale independence, and independence of irrelevant alternatives [19].

Define $J = \{r_{i,j} \mid r \in G, r_{i,j} > L_{i,j}\}$ as the set of VM-pairs that can achieve strictly higher bandwidths compared to their initial rates, i.e., the VM-pair whose bandwidth demand exceeds the base bandwidth. If J is nonempty, then we have the following theorem [16].

Theorem 1. There exists a Nash bargaining solution and the elements of the vector $r = \phi(G, x_0)$ solve the following optimization problem:

$$\max_{r_{i,j}} \prod (r_{i,j} - L_{i,j})^{K_{i,j}}, \forall r_{i,j} \in J. \tag{5}$$

The convex optimization above has a unique solution equivalent to the Nash bargaining solution. Eq. (5) illustrates the form of the joint profit in the bargaining game, which is the product of the utility gains of all the players and can be maximized by the Nash bargaining solution. In particular, the objective function in Eq. (5) is mathematically equivalent to the objective $\max \sum K_{i,j} \ln(r_{i,j} - L_{i,j}), \forall r_{i,j} \in J$.

With the constraints for each $r_{i,j}$,we can obtain the optimization for the fair bandwidth allocation problem (P_r) as follows

$$\begin{aligned}
\max_{r_{i,j}} \quad & \sum_j \sum_i K_{i,j}\ \ln(r_{i,j} - L_{i,j}), \forall r_{i,j} \in J && (6)\\
\text{s.t.} \quad & r_{i,j} \geq L_{i,j},\ \forall i,j \in \mathcal{N} && (7)\\
& r_{i,j} \leq U_{i,j},\ \forall i,j \in \mathcal{N} && (8)\\
& \sum_{i \in V_m} r_i^I \leq C_m\ \forall m \in \mathcal{M} && (9)\\
& \sum_{i \in V_m} r_i^E \leq C_m,\ \forall m \in \mathcal{M}, && (10)
\end{aligned}$$

where $r_i^I = \sum_{j=1}^N r_{j,i}$ and $r_i^E = \sum_{j=1}^N r_{i,j}$ are the ingress and egress rate of VM i, respectively. Eq. (9) and Eq. (10) represent the constraint of the bandwidth capacity for each server. Note that the optimization is a variant of the maximization problem with logarithmic utility in [17], under specific upper and lower bound constraints. The utility function is a special case of the α-fair utility function [20], which ensures the solution vector to be proportionally fair.

For simplicity, we assume $r_{i,j} > L_{i,j}, \forall i,j \in \mathcal{N}$ when characterizing the optimal solution. In fact, for such rates satisfying $r_{i,j} = L_{i,j}$, we can exclude them from the corresponding bandwidth capacities of servers by reducing the server capacity with $L_{i,j}$. Hence, this assumption has no impact on the solution to the optimal problem.

IV. Solution via a Bargaining Approach

In this section, a bargaining game approach is used to construct iterations which converge to the multipliers that solve the optimal rate for each VM-pair. We present the design of *Falloc* based on this game-theoretic approach.

A. Characterizing the Optimal Solution

Given the optimization problem for bandwidth allocation, we first characterize the optimal solution, i.e., the rate for each VM-pair. We use a matrix $W = (w_{m,i})_{M \times N}$ to denote the placement of VMs on each server, where $w_{m,i}$ is a binary variable defined as:

$$w_{m,i} = \begin{cases} 1, & \text{VM } i \text{ is on server } m; \\ 0, & \text{otherwise.} \end{cases}$$

Let $r^I = (r_1^I, r_2^I, \ldots, r_N^I)$ ($r^E = (r_1^E, r_2^E, \ldots, r_N^E)$) be the vector of ingress (egress) rates of N VMs and $C = (C_1, \ldots, C_M)$ be the vector of bandwidth capacity of M servers.

Note that the constraints of the variable r are linear, we can apply the method of Lagrange multipliers, and the KKT conditions [21] are both necessary and sufficient for an existing optimal solution.

Theorem 2. There exists $\lambda_m^I \geq 0$ ($m \in \mathcal{M}$) and $\lambda_m^E \geq 0$ ($m \in \mathcal{M}$) such that for all $i, j \in \mathcal{N}$:

$$r_{i,j}^* = L_{i,j} + \frac{K_{i,j}}{\sum_{m=1}^{M} \lambda_m^E w_{m,i} + \sum_{m=1}^{M} \lambda_m^I w_{m,j}}, \qquad (11)$$
$$L_{i,j} \leq r_{i,j} \leq U_{i,j},$$

where $r_{i,j}^*$ is the unique Nash bargaining solution to the problem (P_r).

Proof: We begin with an assumption that the allocation space $X \in \mathcal{R}^{N \times N}$ is a nonempty, convex and compact set. We define

$$\theta(r) = \sum_{j=1}^{N} \sum_{i=1}^{N} \ln(r_{i,j} - L_{i,j}),$$

then $\theta(\cdot) : X \to \mathcal{R}^+$ is strictly concave [16].

Let $\alpha_{i,j} \geq 0, \forall i, j \in \mathcal{N}$, $\beta_{i,j} \geq 0, \forall i, j \in \mathcal{N}$ and $\lambda_m^I, \lambda_m^E \geq 0, \forall m \in \mathcal{M}$ denote the Lagrange multipliers for the constraint of minimum bandwidth (Eq. (7)), upper-bound bandwidth (Eq. (8)), and server capacity (Eq. (9) and Eq. (10)), respectively. Then, the Lagrangian of the problem is

$$\mathcal{L}(r, \alpha, \beta, \lambda^I, \lambda^E) = \theta(r) - \sum_{j=1}^{N} \sum_{i=1}^{N} \alpha_{i,j}(L_{i,j} - r_{i,j})$$
$$- \sum_{j=1}^{N} \sum_{i=1}^{N} \beta_{i,j}(r_{i,j} - U_{i,j}) - \sum_{m=1}^{M} \lambda_m^I \big((W \cdot r^I)_m - (C)_m\big)$$
$$- \sum_{m=1}^{M} \lambda_m^E \big((W \cdot r^E)_m - (C)_m\big).$$

Giving the necessary and sufficient conditions for the Kuhn-Tucher conditions:

$$\nabla \mathcal{L}(r^*, \alpha, \beta, \lambda^I, \lambda^E) = 0 \Longleftrightarrow$$

$$\frac{1}{r_{i,j}^* - L_{i,j}} + \alpha_{i,j} - \beta_{i,j} - \sum_{m=1}^{M} \lambda_m^I w_{mj} - \sum_{m=1}^{M} \lambda_m^E w_{mi} = 0, \qquad (12)$$

$\forall i, j \in \mathcal{N}$ and

$$\begin{cases} \alpha_{i,j}(L_{i,j} - r_{i,j}^*) = 0, & i, j \in \mathcal{N}, \\ \beta_{i,j}(r_{i,j}^* - U_{i,j}) = 0, & i, j \in \mathcal{N}, \\ \lambda_m^I \big[(W \cdot r^I)_m - (C)_m\big] = 0, & m \in \mathcal{M}, \\ \lambda_m^E \big[(W \cdot r^E)_m - (C)_m\big] = 0, & m \in \mathcal{M}, \end{cases} \qquad (13)$$

where $r_{i,j}^*$ is the optimal solution to the problem (P_r).

To derive the solution, we first consider the values of the multipliers in Eq. (13). For the constraints $r_{i,j} \geq L_{i,j}$ and $r_{i,j} \leq U_{i,j}$, the borderline value is the special case of the allocated rate $r_{i,j}$, hence, we focus on the situation that $L_{i,j} < r_{i,j} < U_{i,j}$, and have $\alpha_{i,j} = 0$ and $\beta_{i,j} = 0$. Then we can obtain $r_{i,j}$ by solving Eq. (12). ■

The original problem in Eq. (6)-(10) is a convex optimization with $2(M + N)$ constraints, whose computational complexity may increase significantly as the number of VMs and servers scales up. Fortunately, the solution in Eq. (11) indicates that each optimal rate $r_{i,j}$ can be solved by the optimal multipliers associated with two servers, i.e., the server hosting the source VM i and the server hosting the destination VM j. Hence, the key to maximize the joint profit in utilizing datacenter networks is to obtain the optimal Lagrange multiplier of each server, which is independent of other servers. This motivates us to obtain the rate of each VM-pair *distributively* rather than using a centralized approach for the optimization.

B. Dual-Based Decomposition

The centralized primal problem in Eq. (6)-(10) can be solved by the dual-based decomposition. Specifically, we first define a primal problem which has the same optimal solution as problem (P_r) and then obtain the dual problem corresponding to the primal problem with no duality gap. The optimal solution to the dual problem can be also derived through the optimal Lagrange multipliers, which can be characterized by the subgradient methods.

First, let us consider the primal problem which has the same optimal solution as problem (P_r). The objective has the following form:

$$\min_{r} \qquad P(r) = -\sum_{j=1}^{N} \sum_{i=1}^{N} K_{i,j} \ln(r_{i,j} - L_{i,j}). \qquad (14)$$

We focus on the case of $L_{i,j} < r_{i,j} < U_{i,j}$. Let X be the allocation space of r as defined in Sec. III-C. The Lagrangian associated with the primal problem in Eq. (14) is defined as $\mathcal{L}(\cdot) : X \times \mathcal{R}^M \times \mathcal{R}^M \to \mathcal{R}$, where

$$\mathcal{L}(r, \lambda^I, \lambda^E) = -\sum_{j=1}^{N} \sum_{i=1}^{N} K_{i,j} \ln(r_{i,j} - L_{i,j})$$
$$+ \sum_{m=1}^{M} \lambda_m^I \big((W \cdot r^I)_m - C_m\big) + \sum_{m=1}^{M} \lambda_m^E \big((W \cdot r^E)_m - C_m\big).$$

Note that λ^I and λ^E are the dual variables associated with the problem. The Lagrange dual function $d(\cdot) : \mathcal{R}^M \times \mathcal{R}^M \to \mathcal{R}$ corresponding to $\mathcal{L}(r, \lambda^I, \lambda^E)$ is expressed as

$$d(\lambda^I, \lambda^E) = \inf_{r \in \mathcal{R}^{N \times N}} \mathcal{L}(r, \lambda^I, \lambda^E).$$

Since the primal problem has a unique optimal solution, the dual function yields lower bounds on the each optimal $r_{i,j}^*$ which solves Eq. (14). For any $\lambda^I, \lambda^E \in \mathcal{R}^M$, we have $d(\lambda^I, \lambda^E) \leq \mathcal{L}^*$, where $\mathcal{L}^*$ is the infimum of $\mathcal{L}(r, \lambda^I, \lambda^E)$. The infimum of Lagrangian occurs where the gradient is equal to zero, thus $r^*_{i,j} = L_{i,j} + K_{i,j}/(\sum_{m=1}^{M} \lambda^E_m w_{mi} + \sum_{m=1}^{M} \lambda^I_m w_{mj})$ according to Theorem 2.

It is clear that there exists r such that for each $m \in \mathcal{M}$, $(W \cdot r)_m < C_m$ and $L_{i,j} < r_{i,j} < U_{i,j}$. This implies that there exists r in the relative interior of the intersection of the domain of all constraint functions. Because X is convex and $P(r)$ is convex over X, the Slater's condition holds, which is a sufficient condition for strong duality [21]. Hence, there is no duality gap and there exist λ^I and λ^E satisfying $d(\lambda) = \mathcal{L}^*$.

To conclude, we obtain the dual problem corresponding to the primal problem with no duality gap. The dual problem (P_d) is described as follows:

$$\max_{\lambda^I, \lambda^E \in \mathcal{R}^M} d(\lambda^I, \lambda^E) = \mathcal{L}(r^*, \lambda^I, \lambda^E), \tag{15}$$

where $d(\lambda^I, \lambda^E)$ is the dual function and $\mathcal{L}(\cdot)$ is the Lagrangian of the primal problem.

C. Subgradient Methods for Dual Problem

To solve the primal problem, we first obtain the optimal solution to the dual problem. By using a suitable step-size, we design an iteration that converges to the optimal λ^I (λ^E) by applying the subgradient methods [21]. Since the strong duality holds as discussed above, we can achieve the optimal Nash bargaining solution with Eq. (11).

Let the set $\overline{\lambda}$ denote the optimal solution to the dual problem and the set $\overline{\mathcal{R}}$ be the solution to the primal one. We define the following recursion based on [20]:

$$\lambda_m^{(s+1)} = \max(0, \lambda_m^{(s)} + \xi \frac{\partial d}{\partial \lambda_m}), \forall m \in \mathcal{M}, \tag{16}$$

where ξ is the step-size. We first discuss the sequence of λ^I, while regarding λ^E as a constant.

It has been proved in [21] that $(\lambda^I)^{(s)}$ converges in $\overline{\lambda}$ as $s \to \infty$ if we choose such a step size according to the following condition.

Proposition 1. For the recursive sequence $\{(\lambda^I)^{(s)}\}$, if $(\lambda^I)^{(0)} \in \mathcal{R}^{+M}$ and ξ satisfy the *diminishing step size rules* [21], then the recursion of dual variable $\{(\lambda^I)^{(s)}\}$ converges, thus

$$\lim_{s\to\infty} (\lambda^I)^{(s)} = \lambda^{I*} \in \overline{\lambda}. \tag{17}$$

Having determined the step size in Eq. (16), the gradient of the dual function can also be solved by obtaining the partial derivatives of $d(\lambda^I, \lambda^E)$ as below

$$\frac{\partial d(\lambda^I, \lambda^E)}{\partial \lambda^I_m} = \sum_{j=1}^{N} w_{m,j} \sum_{i=1}^{N} r^*_{i,j} - C_m. \tag{18}$$

For the sequence $\{(\lambda^E)^{(s)}\}$, we have the similar conclusion and the partial differential is

$$\frac{\partial d(\lambda^I, \lambda^E)}{\partial \lambda^E_m} = \sum_{i=1}^{N} w_{m,i} \sum_{j=1}^{N} r^*_{i,j} - C_m. \tag{19}$$

In Theorem 2, we have obtained the explicit form of optimal rate $r^*_{i,j}$. The sequences generated by Eq. (16) converge to the optimal solution to the dual problem (P_d) in Eq. (15) according to Proposition 1. Since there is no duality gap in the dual decomposition, the rate vector r associated with λ^I and λ^E converges to the Nash bargaining solution, thus $\forall i, j \in \mathcal{N}$

$$\lim_{s\to\infty} r((\lambda^I)^{(s)}, (\lambda^E)^{(s)}) = r^* \in \overline{R}. \tag{20}$$

In summary, the approach to obtain the optimal rate can be viewed as an iterative bargaining process, where the VM-pairs (buyers) bargaining for bandwidth with the servers (sellers). The dual variables serve as the bargaining prices (Eq. (16)) and the rates indicate the utility gains of the VM-pairs (Eq. (11)). For example, $r_{i,j}$ is the utility of VM-pair from i to j. Suppose VM i is hosted on server m and VM j is hosted on server l, then the dual variables of server m and server l are λ^E_m and λ^I_l, respectively. If server m has any remaining bandwidth, it cuts down the price of the bandwidth.

Then the VM-pair will buy more bandwidth in the next round which is indicated by the increase of rate $r_{i,j}$ as the dual variables decrease. Therefore, the remaining bandwidth can be allocated to the VM-pairs with unsatisfied bandwidth demand, and the network utilization increases. Otherwise, when the total allocated rates exceed the capacity of server m, the price will increase so as to reduce the excessively allocated bandwidth.

V. Algorithm Design

A. Offline Cooperative Algorithm

Based on the game-theoretical framework, we present the bandwidth allocation algorithm for datacenter networks in this section. We focus on two interesting results in the bargaining game approach: (i) the dual variables (λ^E_m and λ^I_m) of each server m can be updated independently with local information in Eq. (19). (ii) the iteration of each rate $(r_{i,j})$ in Eq. (11) only requires the bandwidth information of the server hosting VM i and the server hosting VM j. This motivates the design of *Falloc*, which can obtain the optimal rate in a distributed cooperative manner.

Update weight and base bandwidth. We first consider how to update the base bandwidth and weight of each VM-pair. As Eq. (1) and (2) shows, when a VM establishes a connection or disconnects with another VM, the number of its connected VMs changes. Thus the base bandwidth and weight of existing VM-pairs should be changed accordingly. The updating process is shown in Algorithm 1. The function is triggered by the change of connections to a VM: when it connects to or disconnects from another VM x, it obtains the weight and base bandwidth of VM x, and then updates the weight and base bandwidth of all its VM-pairs. To reduce the overhead of re-calculation, each VM-pair can be managed by its source VM.

The outputs of Algorithm 1, i.e., the base bandwidth and weight, are used as input for calculating the optimal rate allocation for VM-pairs.

Distributed iteration process. Since the calculation of rate $r_{i,j}$ for each VM-pair involves two servers, it can be executed either on the source server or the destination server. We present the iteration process of each server in Algorithm 2.

Algorithm 1 Update weight and base bandwidth

```
Input:
  The base bandwidth of VMs $< B_i^I, B_i^E >, \forall i \in \mathcal{N}$
  The weight of VMs $< K_i >, \forall i \in \mathcal{N}$
Output: Lower bound $L_{i,j}$ and upper bound $U_{i,j}$.
  if VM $i$ connects/disconnects a VM then
    for all VM-pair $i, j$ do
      Update $B_{i,j}$ as Eq. (1) and $K_{i,j}$ as Eq. (2)
      Initialize $L_{i,j}$ as Eq. (3) and $U_{i,j}$ as Eq. (4)
    end for
  end if
```

Let $r_{p_m}^E = \sum_{i=1}^N (w_{m,i} \sum_{j=1}^N r_{i,j}) - C_m$ and $r_{p_m}^I = \sum_{i=1}^N (w_{m,i} \sum_{j=1}^N) r_{i,j} - C_m$ represent the allocated egress and ingress bandwidth of server m. According to Eq. (3), for the VM-pair whose bandwidth demand ($D_{i,j}$) is less than the base bandwidth ($B_{i,j}$), the algorithm sets $r_{i,j}$ to be $D_{i,j}$. For other VM-pairs which belong to J in Sec. III-C, the iteration of $r_{i,j}$ begins with the initial lower bound $L_{i,j}$. During each iteration, server m updates its dual variables (λ_m^E and λ_m^I) by calculating its residual bandwidth ($r_{p_m}^E$ and $r_{p_m}^I$) according to Eq. (16). For all the VM-pairs ($r_{i,j}$) with VM i on server m, the server should request the other dual variable λ_l^I from server l ($j \in V_l$). The rate of the VM-pair is then updated based on these two dual variables according to Eq. (11). In addition, we make a judgment in case that the rate exceeds the upper bound bandwidth $U_{i,j}$. The step size is chosen according to the step size rules in Proposition 1 and updated locally within each iteration.

Stopping Rules. Note that the convergence speed of $r_{i,j}$ depends on the step-size ξ and the gradient of the dual variables, which are exactly the residual bandwidth of server m and l. Since the step-size and residual bandwidth are both maximal initially and decrease as the algorithm performs, the dual variables will quickly converge to approximate optimal values and then slowly approach the optimal values. Hence, the algorithm can finish within an acceptable number of steps if we do not need strict optimal rates for all VM-pairs. We can define two stopping rules in *Falloc* to balance the tradeoff between algorithm overhead and precision.

- *Step-based mechanism:* Use S as the total iteration steps in the convergence process. As the execution time of the algorithm is in proportion to the iteration rounds, one can manage the cost of the algorithm by adjusting the iteration rounds. For example, if the cloud provider cares much about the algorithm's overhead, he can choose a small S (e.g., $S < 50$) in this mechanism to reduce the execution time.
- *Precision-based mechanism:* Stop the convergence process if the variation of each $r_{i,j}$ is less than Δ within two consecutive iterations. For example, if the cloud provider has a SLA on bandwidth allocation for VMs with the tenant, he will need to specify a small Δ (e.g., $\Delta = 0.1$ Mbps) so as to fulfill his agreement. The precision of the output is also closely related with the iteration steps, as an accurate bandwidth allocation can cost much more steps than a rough one.

When the iteration stops with the control of the stopping rules, *Falloc*'s outputs can be applied in the hypervisors by enforcing a bandwidth limitation for each VM-pair. The allocated bandwidth, which is always less than (or equal to) the bandwidth demand, will be fully utilized due to the aggressiveness of transport-layer flows.

Algorithm 2 Bandwidth allocation on server m

```
Input:
    The step-size: $\xi$
    Server bandwidth capacity: $C_m, \forall m \in \mathcal{M}$
    Bandwidth demand matrix: $[D_{i,j}]_{N \times N}, \forall i \in \mathcal{N}$
    VM placement: $[w_{m,i}]_{M \times N}, \forall m \in \mathcal{M}, \forall i \in \mathcal{N}$
    The total number of iteration rounds: $S$
    The gap between two consecutive iterations: $\Delta$
Output:
 1: while $s < S$ or $r_{i,j}^{(s)} - r_{i,j}^{(s-1)} > \Delta$ do
 2:   Update allocated bandwidth
      $r_{p_m}^E = \sum_i w_{m,i} r_i^E$, $r_{p_m}^I = \sum_i w_{m,i} r_i^I$
 3:   Update dual variables as Eq. (16)
      $\lambda_m^E = \max(0, \lambda_m^E - \xi(C_m - r_{p_m}^E))$
      $\lambda_m^I = \max(0, \lambda_m^I - \xi(C_m - r_{p_m}^I))$
 4:   for all $r_{i,j}, i \in V_m$ do
 5:     Update $\lambda^E = \lambda_m^E$
 6:     Obtain $\lambda^I = \lambda_l^I$ from server $l$, $j \in V_l$
 7:     if $\frac{K_{i,j}}{\lambda^E + \lambda^I} > U_{i,,j} - L_{i,j}$ then
 8:       $r_{i,j}^{(s)} = U_{i,j}$
 9:     else
10:       $r_{i,j}^{(s)} = L_{i,j} + \frac{K_{i,j}}{\lambda^E + \lambda^I}$
11:     end if
12:   end for
13:   Update step size $\xi$
14:   Update iteration round $s = s + 1$
15: end while
```

B. Online Algorithm and Message Exchange

In the offline algorithm, the bandwidth demand of each VM is assumed given. However, existing works (e.g., [12], [22]) have shown that datacenter traffic is highly dynamic, making it hard to accurately predict the bandwidth demand between VMs. As a consequence, the allocated bandwidth may not be fully utilized, and the residual bandwidth ($r_{p_m}^E$ and $r_{p_m}^I$) and actual rate of VM-pairs can hardly be calculated.

To practically implement *Falloc* in datacenter networks, we propose to develop an online algorithm that can dynamically enforce rate allocation without requiring the traffic demand of each VM. Based on the offline algorithm, the design consists of two components: the bandwidth allocation algorithm on each server (Algorithm 3) and the communication protocols (Fig. 3) between servers.

Bandwidth allocation algorithm. There are three main differences between the online algorithm and the optimal algorithm. First, the rate of server is obtained by monitoring the traffic of network interface, which represents the real-time rate of a server. The observed rate of a local server in

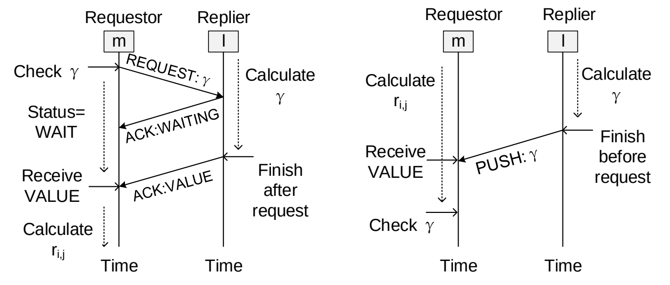
Algorithm 3 is equal to the calculated rate in Algorithm 1, and can be monitored by a agent program on each server. As packets in datacenter networks have low RTT (which is in the order of 1ms), the information of rate received from a remote server in Algorithm 3 can be viewed as the real-time rate of that server. This way, there is no need to calculate the rate using bandwidth demand of VMs. Second, the rate of VM-pair is directly enforced as a rate-limit to shape the traffic between VMs during each iteration, rather than applied after the convergence. Since the rate-limit is the upper-bound rate of each VM-pair, the actual rate will be the minimum one of the rate-limit and bandwidth demand. This way, the algorithm avoids the need to estimate the actual rates with the bandwidth demand of VM-pairs, and we have real-time rate allocation as it is performed within each iteration. Third, for VM-pairs whose actual rates are less than their respective base bandwidths, the rate-limit will be $B_{i,j}$. Once $B_{i,j}$ is fully utilized, the algorithm will regard $r_{i,j}$ as the VM-pair whose bandwidth demand exceeds the base bandwidth (i.e., $r_{i,j} \in J$), thus allocating a portion of the residual bandwidth to $r_{i,j}$ in the next round.

Algorithm 3 Online bandwidth allocation

```
Input:
    The step-size: $\xi$
    Server bandwidth capacity: $C_m, \forall m \in \mathcal{M}$
    Bandwidth demand matrix: $[D_{i,j}]_{N \times N}$, $\forall i \in \mathcal{N}$
    VM placement: $[w_{m,i}]_{M \times N}$, $\forall m \in \mathcal{M}, \forall i \in \mathcal{N}$
Output:
 1: Obtain the dual variable of server $m$: $\lambda_m^I$
 2: Update the local dual variable as Eq. (16)
    $\lambda_m^E = \max(0, \lambda_m^E - \xi(C_m - r_m^E))$
 3: for all $r_{i,j}, i \in V_m$ do
 4:     Obtain the actual rate $r_{i,j}$
 5:     if $r_{i,j} < B_{i,j}$ then
 6:         $r_{i,j} = B_{i,j}$
 7:     else
 8:         Update $\lambda^E = \lambda_m^E$
 9:         Obtain $\lambda^I = \lambda_l^I$ from server $l$, $j \in V_l$
10:         $r_{i,j} = B_{i,j} + \frac{K_{i,j}}{\lambda^E + \lambda^I}$
11:         Enforce rate-limit $r_{i,j}$ of VM-pair $i$ to $j$
12:     end if
13: end for
```

Message exchange protocols. Since the servers need to share the dual variables with their connected neighbors and require input data from the centralized controller, *Falloc* defines the protocols for such communication, aiming at minimizing the delay and overhead of message exchange. We describe the message exchange protocols under two conditions.

Fig. 3 shows the message exchange between two iteration servers. Each server in the communication plays two different roles: requestor and replier. When a *request* server wants to request a dual variable (λ_l^I), it first checks if it has received the dual variable. If it has the dual variable (Fig. 3(b)), it will continue the iteration with this value. Otherwise (Fig. 3(a)), it sends a REQUEST message to the paired server (j,$l \in V_j$) and stays in WAIT status until it receives an ACK with VALUE of



(a) Request mode: request happens before the dual variable is ready

(b) Push mode: the dual variable is pushed before it is requested

Fig. 3. Message exchange protocols between request server and reply server.

the requested dual variable. After receiving the REUEST, the reply server (j) will then send back an ACK of WAITING if the requested dual variable is not ready and resends an ACK consisting of the dual variable until it finishes the calculation.

On the other hand, when a *reply* server finishes calculating a new dual variable, it will first check the REQUESTs for this variable. If there are REQUESTs (Fig. 3(a)), it sends ACKs with this dual variable to the request servers. Otherwise (Fig. 3(b)), it pushes the dual variable to all the paired servers for future usage.

Falloc uses a centralized controller to manage the information of VMs (e.g., base bandwidth, weight and bandwidth demand). To implement *Falloc* in datacenters, we also need to specify the communication schemes for message exchange between controllers and hosting servers. The iteration process can be performed in a distributed manner on each server. When the iteration server receives a START command, it will request the input data (as shown in Algorithm 2) from the controller and begin to run bandwidth allocation algorithm. This distributed solution only has a computational complexity of $S \cdot O(n)$ on each server, but transmits about $S \cdot M$ messages in total.

C. Falloc Implementation

In this paper, we implement *Falloc* with the distributed solution as described in Sec. V-A. The *Falloc* prototype runs our proposed bandwidth allocation algorithm at each sender server, and enforces the allocation by capping the rate of VM-pairs. Specifically, it consists of two basic components:

- **Bandwidth allocation service.** The service is responsible for updating the base bandwidth/weight of VM-pairs, and calculating the rates to be allocated. On one hand, when the connections of VMs change, it updates the base bandwidth and the weight of VM-pairs according to Algorithm 1. On the other hand, it reads the input data following the message exchange protocols, and then performs Algorithm 2 to obtain the optimal rate for each VM-pair. After obtaining the rate, it notifies the rate controller to set up rate-limit for specific VM-pairs.
- **Rate controller.** The rate controller is a program running on the sender servers, which manages the rates between VM-pairs. We implement it with Traffic Control (TC) tool in Linux based OS. When the controller receives

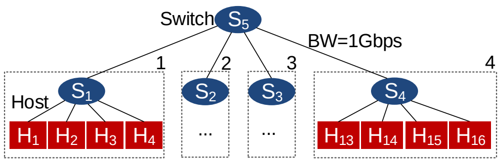


Fig. 4. Testbed with 16 hosts divided into 4 groups. Each group consists of 4 hosts and 1 switch. One group is equivalent to a server.

commands from the bandwidth allocation service, it immediately sets up the rate-limits for VM-pairs to shape the traffic on this server.

VI. Performance Evaluation

We evaluate the performance of *Falloc* from two following aspects:

- First, we validate that *Falloc* can achieve bandwidth guarantee, proportional sharing and work-conserving under aggressive bandwidth competition by carrying out experiments on a real testbed. We also qualify the rate of convergence for the algorithms and the impact of base bandwidth on the bandwidth allocation.
- Second, we analyze the network utilization, overhead of algorithms and the job completion time through large scale simulations with Mapreduce workload traces. We also examine the impact of the precision of the algorithms on job completion times.

A. Experimental Results

We evaluate *Falloc* prototype in Mininet, an evaluation platform running real network protocols and workloads. Our first set of experiments is to quantify the bandwidth allocation of *Falloc*, by characterizing the impact of different base bandwidth B and weight K with specified bandwidth demand for VMs. As shown in Fig. 4, we build a testbed consisting of 16 hosts, and every 4 hosts are connected to a switch, equivalent to a server with 4 VMs. The edge switches are connected to a single root switch with 1Gbps bandwidth. The workload is constructed by generating network traffic based on the specified demand matrix D_N. In the experiment, we focus on how the protocol allocates the bandwidth on a congested link, by showing the allocated rates of H_1, H_2, H_3 and H_4 under typical scenarios.

Bandwidth guarantee and utilization. In this experiment, the base bandwidth and the weight of each VM are set to 250Mbps and 1, respectively. Each VM sends out data to a respective remote server, where H_4 is aggressive with 10 TCP links and others have 1 TCP link. As shown in Fig. 5, without guarantees, H_4 aggressively takes up most of the bandwidth, and causes unfairness to other VMs. On the contrary, *Falloc* can guarantee sufficient bandwidth for VM H_1, whose bandwidth demand is less than the base bandwidth. For VM H_2, H_3 and H_4, whose bandwidth demands exceed the base bandwidths, *Falloc* fairly allocates the bandwidth to each VM (each has about 300Mbps), as they have the same base bandwidth and weight. Fig. 5 reveals that by limiting the traffic at the application layer, *Falloc* can provide

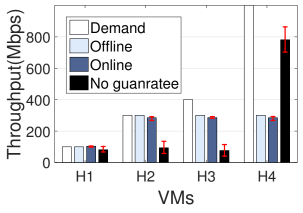


Fig. 5. Throughputs of VMs with different bandwidth demands, while B of each VM is 250 Mbps.

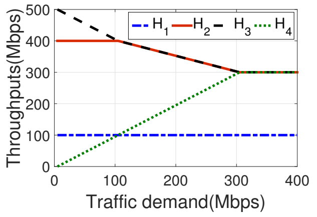


Fig. 6. Throughputs of VMs with increasing traffic demand of VM H_4.

TABLE II
Throughput and Weight of VMs with $B = 0$

VM (Mbps)	H_1	H_2	H_3	H_4
Sum of weights of VMs	2	3	4	5
Throughput of VMs (Mbps)	141.3	211.3	277.1	342.2
Ratio of throughput	2	2.99	3.92	4.84

bandwidth guarantees for VMs irrespective of the competition of aggressive VMs. In this case, the gap of rates between the online algorithm and the offline algorithm is less than 6%. In the following experiments, we use the results from the offline algorithm as *Falloc*'s outputs, as the offline and online algorithms have similar output and the offline algorithm represents our expected bandwidth allocation.

To show the bandwidth competition among VMs, we increase the bandwidth demand of VM H_4 from 0 to 400Mbps. Fig. 6 shows the throughput of these VMs under *Falloc*'s bandwidth allocation. The throughput of H_1 stays static, since its rate is under the base bandwidth, and the algorithm guarantees the bandwidth for H_1. The increase of H_4's throughput is at the cost of reducing the rates of VMs that exceed the base bandwidths. The rate of H_3 is firstly reduced as it is the maximal one, and when it reaches the same rate with H_2, they both decrease until H_2, H_3 and H_4 have the same throughput. This is because all these VMs share the same weight, and algorithm maintains fairness among them.

Impact of weight. We characterize the impact of weight K by assigning different weights and the same base bandwidth B to the VMs on server 1. Fig. 7 plots the bandwidth allocation for $H_1 \sim H_4$ with different settings of base bandwidth, i.e., $B = 0, 150, 250$ Mbps for each VM, and all the VMs have infinite bandwidth demand (represented by a value larger than the physical bandwidth). On the one hand, Fig. 7 shows that *Falloc* guarantees the base bandwidth of VMs regardless of the weights of other VMs. Under each setting, the observed rate of each VM is larger than B. Particularly, when the base bandwidth is a full partition of the physical bandwidth, i.e., $B = 250$ Mbps, the weight will have no effect on the allocation result, since there is no residual bandwidth to be shared after guaranteeing the bandwidth demand for each VM. Note that the difference of bandwidth allocations is caused by the error in rate enforcement.

On the other hand, when the base bandwidth equals to zero, *Falloc* will not guarantee bandwidth and only shares the bandwidth proportionally among different tenants (a group of connected VMs). This is showed by the proportion of the

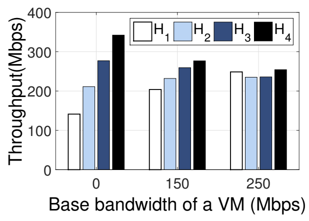


Fig. 7. Throughputs of VMs with varying the base bandwidth, when $K_1 : K_2 : K_3 : K_4 = 1 : 2 : 3 : 4$.

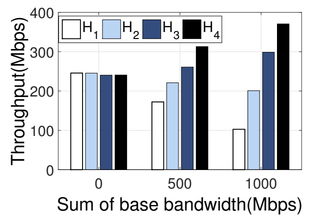


Fig. 8. Throughputs of VMs with different base bandwidth for each VM.

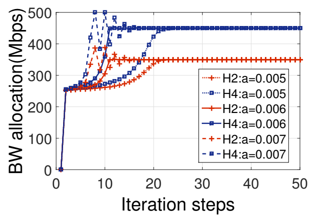


Fig. 9. Rate variation of VM H_2 and H_4 with increasing number of iteration steps.

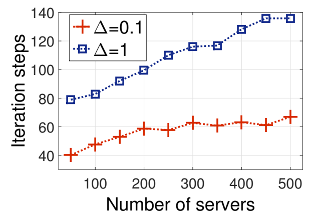
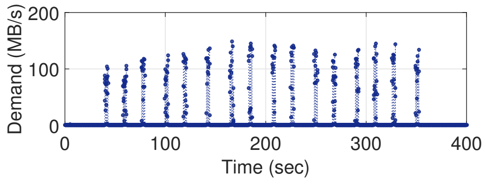
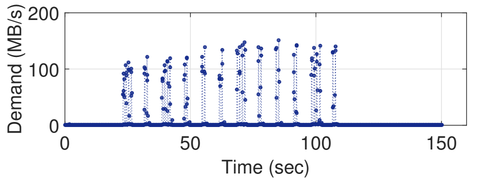


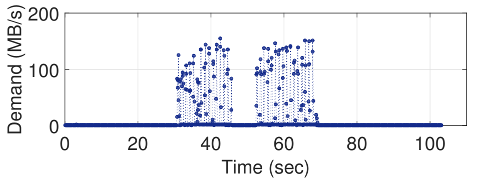
Fig. 10. Average convergence steps with increasing number of servers when $\Delta = 1$ and 0.1 Mbps.



(a) Hadoop Word Count with 1.2 GB input data.



(b) Hadoop Sort with 1.0 GB input data



(c) Hadoop Join with 1.0 GB input data

Fig. 11. Network throughput of VM when running Mapreduce workloads.

rates with $B = 0$ in Table II. There are 4 groups of VMs, each of $H_1 \sim H_4$ connecting to one remote VM. The weight ratio of the group with $H_1 \sim H_4$ is $2 : 3 : 4 : 5$. Table II shows the throughput of VMs in different groups, which is shaped proportionally with the ratio $2 : 2.99 : 3.92 : 4.84$. We find that the rates of VMs with higher throughput have larger fluctuation with our rate-limit tools. As a result, the VMs with larger weights suffer performance degradations in the bandwidth allocation, and the ratio of throughput is not strictly equal to the weight ratio. Nevertheless, the error is acceptable and it is worthwhile to use the weight to classify the applications with different priorities.

Impact of base bandwidth. The base bandwidth B is another basic metric that determines the bandwidth allocation. To validate the impact of B, we set up 3 experiments with the same weight K, and assign different B to $H_1 \sim H_4$, while maintaining the proportion as $1 : 2 : 3 : 4$. As shown in Fig. 8, when each B is 0, the bandwidth is equally allocated due to the same weight of all the VMs. However, when the sum of B is maximized, i.e., equals the 1000 Mbps physical bandwidth, the proportion of bandwidth for $H_1 : H_2 : H_3 : H_4$ is strictly $1 : 2 : 3 : 4$, though they are assigned with the same weight. This is because the *Falloc* protocol can guarantee the bandwidth demand if it is less than the base bandwidth.

Balance the tradeoff. The above analysis for Fig. 7 and Fig. 8 verifies the tradeoff between bandwidth guarantee and proportional bandwidth share. In our solution, we choose to give priority to bandwidth guarantee and consider proportional sharing as a complement. Experiments show that by changing the base bandwidth, *Falloc* can balance this tradeoff, and obtain an allocation involving both bandwidth guarantee and proportional share for VMs. This is a good start to provide flexible fairness in sharing datacenter networks and by using proper B and K based on a rigorous optimization, one can assign suitable bandwidth to different applications towards maximizing the datacenter's performance.

Rate of convergence. We now quantify the convergence by showing the rates of the VMs in the iteration process. We give the result in the scenario corresponding to the experiment shown in Fig. 5. The step size ξ is set to a/S and the rates of VM H_1 and H_3 are omitted since they remain unchanged during the iteration. As we can see in Fig. 9, the rates of VMs converge under the control of our allocation algorithm. We find that the optimal step size should be chosen according to the problem scale. A large step size may lead to fluctuation during the convergence, while a small step size will slow down the convergence speed.

To verify the convergence speed in general cases, we randomly generate 10 groups of demand matrices with a varying number of servers from 50 to 500. Each group consists of 50 demand matrices and the bandwidth demand of each VM-pair is subject to the uniform distribution $U(0, 1000)$ in Mbps. We assert that the algorithm converges when the variation of every $r_{i,j}$ in the rate matrix is less than Δ within one iteration. Fig. 10 shows the average convergence steps with $\Delta = 1$ and 0.1 Mbps. Allowing the error to be as much as 1 Mbps, our algorithm can converge to a suboptimal bandwidth allocation with less than 65 steps within 1 second even when the number of servers arrives at 500, which should be considered satisfactory.

B. Large Scale Simulations

To evaluate how *Falloc* performs under scenarios with dynamic bandwidth demand on a large scale, we implement *Falloc* with a real-world trace based simulator in C++. The implementation simulates the behavior of a Hadoop cluster in the cloud by analyzing the workload traces of Mapreduce

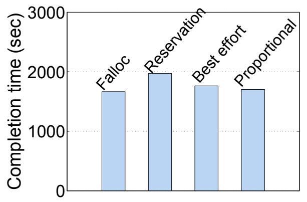


Fig. 12. Comparing the completion time between *Falloc* and other sharing policies.

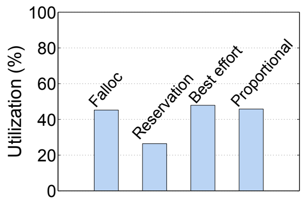


Fig. 13. Comparing the average network utilization between *Falloc* and other sharing policies.

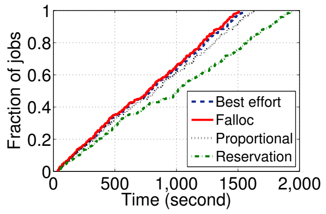


Fig. 14. Fraction of job completion time for *Falloc* and other sharing policies.

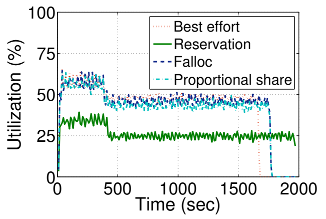


Fig. 15. Comparing network utilization of the entire datacenter during the execution of all batched jobs.

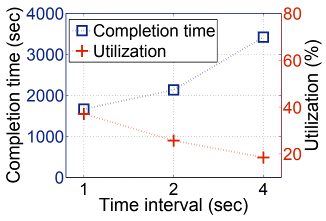


Fig. 16. Job completion time and average network utilization with different time $\Delta t = 1, 2, 4$ s.

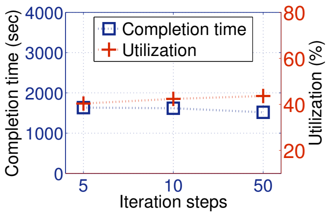


Fig. 17. Job completion time and average network utilization with different iteration steps $S = 5, 10, 50$.

jobs. Specifically, we consider the bandwidth consumption of reading/writing HDFS and data transmission in shuffle phase, as well as the computation time in map/reduce phase.

Characterizing Mapreduce workload. We characterize the network throughput of map/reduce tasks by using two VMs located in the same server to execute the map task and reduce task, respectively. The measurement is conducted in Hadoop 1.0.0 platform, with several typical Mapreduce jobs as the workloads. The server has two 4-core Xeon 2.4 Ghz CPUs and 32 GB memory, running KVM based virtual machines, and each VM is allocated with one core and 4 GB memory. Fig. 11(a)-11(c) show the instantaneous network throughput of the Mapreduce jobs (Hadoop Word Count, Sort and Join) with a 100 ms time interval. Since we do not set limitation to the bandwidth between these two co-located VMs, the capped rates of shuffling indicate that the bottleneck is not in network bandwidth. Based on the observation in [23], we can use these capped rates as the bandwidth demand of the tasks in Mapreduce jobs.

We consider a multi-tenant datacenter with homogeneous servers, which have equal ingress and egress network bandwidth of 1 Gbps. Jobs in the datacenter are running in VMs and each server has 2 VM slots. The simulations use a full bisection bandwidth network following [4]. The simulator simulates a local area (managed by one centralized controller) consisting of 200 servers in the datacenter, where batched jobs are all submitted at one time. A job's tasks are scheduled to run if there are available map and reduce slots in the datacenter, and the job size is represented by the number of VMs needed by this job. For each simulation, we generate 200 mixed jobs, and the job size is exponentially distributed around a mean of 49 (as [5]). Since our policy do not consider VM placement, it is unnecessary to directly compare the performance of *Falloc* with other VM allocation based policies. On the contrary, we assume that the VM allocation has been done before applying our policy. We investigate *Falloc*'s performance with comparison to three commonly used bandwidth sharing policies: (i) best effort: no application-layer bandwidth allocation, (ii) static reservation: the bandwidth of each VM is static, (iii) proportional share: bandwidth is shared in proportion to each VM's weight. We apply equivalent base bandwidth and weight for each VM, i.e., $B = 250$ Mbps, $K = 1$.

Job completion time. The simulator runs the same batched jobs with each sharing policy at a time to compare the job completion time. The number of iteration steps is set to 50 and *Falloc* updates the allocation every one second. The reservation policy statically reserves 250 Mbps bandwidth for each VM, and the proportional share policy uses the average bandwidth requirement as the weight. Fig. 12 shows the simulation result, where *Falloc* reduces the total completion time by about 16% compared to the reservation policy. Since the best effort bandwidth competition and proportional sharing aggressively utilize the available bandwidth, *Falloc* shows no advantage on improving job completion time. However, these two policies do not provide deterministic minimum bandwidth guarantee, and will fail to guarantee predictable performance for applications.

Fig. 14 shows the CDF of completion times of different sharing policies. As jobs are divided into small tasks of equal size, the completion times of different jobs are approximate to each other, thus the fraction of jobs is approximately linear. As we can see, the performance of *Falloc* is as good as the best effort approach while maintaining fairness among different jobs. By making use of the spare bandwidth of jobs that are not producing network traffic, *Falloc* accelerates the completion of Mapreduce jobs, and the advantage increases as the number of job increases.

Network utilization. To unfold the reason for the decrease of the job completion time, we compare the average network utilization among these policies with the same input data. Fig. 13 shows that *Falloc* achieves on average 45.2% network utilization (the realtime utilization is shown in Fig. 15), 18.8% higher than reservation and almost as high as best effort. The improvement in network throughput verifies that *Falloc* can make better use of network resource while providing bandwidth guarantees for VMs.

Precision and overhead. We repeat the above simulations by varying the main factors impacting the cost of running *Falloc*, i.e., the updating time interval Δt and the number of iteration steps S.

As shown in Fig. 16, as Δt varies from 1s to 4s, the completion time increases and the network utilization decreases sharply. The implication is that by extending the time interval, the allocated bandwidth can not quickly adapt to the changing bandwidth demand and the excessively allocated bandwidth will be wasted as the demand reduces. Fig. 17 shows the job completion time and average network utilization with different iteration steps. The iteration rounds within each allocation have little impact on the performance and the job completion times are almost the same, even when we only perform 5 rounds of iteration. This is because *Falloc* can quickly approach the optimal work-conserving bandwidth allocation (as Fig. 9 shows). Note that in the algorithm, the rate-limit for each VM-pair is initially set as the lower bound, which ensures a minimum guarantee based on the bandwidth demand. Hence, *Falloc* can guarantee the performance of VMs without too much iterations.

As a result, to reduce the overhead as well as maintain high performance, a practical method is to reduce the iteration rounds while using small time interval to update the bandwidth allocation.

VII. Related Work

Recently, researchers have observed severe unfairness among VMs caused by sharing networks via TCP in IaaS cloud platforms. To achieve predictable network performance for cloud applications, cloud providers need to maintain fairness at VM or tenant level. Such goals motivate researchers to design new policies and systems for sharing datacenter networks. The proposed mechanisms consist of two main basic ideas, i.e., guaranteeing bandwidth for VMs and proportional sharing bandwidth among VMs or tenants.

Previous works, such as Oktopus [5], SeconNet [8] and [24], focus on providing deterministic bandwidth guarantee for VMs. They allocate VMs into servers based on VMs' bandwidth requirements, and by enforcing bandwidth reservations in both hosting servers and switches, they can ensure the bandwidth of inter-VM network and achieve predictable network performance for the applications in these VMs. The main disadvantage of reservation policies is that they may not be able to achieve high utilization of datacenter networks due to the variation in bandwidth demand of cloud applications. In [23], the authors propose a time varying reservation policy based on the dynamic bandwidth requirements of applications, which increases the utilization of datacenter networks when reserving bandwidth for VMs. However, the policy is only suitable for pulse-like bandwidth demand. Unlike the VM placement based methods above, [13], [25], [26] leverage dynamic end-based rate control for bandwidth guarantee. The bandwidth allocation is work-conserving since the control algorithm at the sender module periodically update the rate-limit, and the unused bandwidth left by one VM can be used by another. Similarly, our work enforces dynamic rate control for VMs. However, our solution uses a cooperative manner that can take advantage of the bandwidth information to avoid fluctuations in rate control.

Other works provide network isolation for VMs by sharing the bandwidth proportionally. Seawall [6] provides a hypervisor based mechanism to slice the bandwidth of each congested link according to weights of source VMs. NetShare [27] allocates the relative bandwidth among different services based on their weights to provide constant proportionality of tenants throughout the network. Faircloud [4] presents understanding on the key requirements and properties for network sharing problem in datacenters. The authors propose three bandwidth allocation policies based on proportional sharing to explore the tradeoff in sharing datacenter networks. These weight based competition mechanisms, however, can not provide deterministic bandwidth guarantee for VMs.

Finally, the game-theoretical framework is built upon previous theoretical basis (e.g., [15], [17], [18]) on using Nash bargaining solution in computer networks. However, different from previous works which focus on providing fairness at flow-level, we propose mechanisms to provide minimum bandwidth guarantee for the network requirements of IaaS datacenters, based on the fairness notion in these works.

VIII. Conclusion

To summarize, we have applied the game-theoretic framework to bandwidth allocation problem in IaaS datacenters. Through a cooperative approach, we present the design of *Falloc* with both offline and online algorithm that solves the optimization of Nash bargaining solution. *Falloc* guarantees the bandwidth requirement based on the base bandwidth for each VM, and shares the residual available bandwidth in a proportional way according to VM's weight. The experiment with prototype implementation shows that *Falloc* can provide flexible fairness for VMs by balancing the tradeoff between bandwidth guarantee and proportional bandwidth share. Our trace-driven simulations show that *Falloc* can achieve high network utilization and good job completion time in datacenter networks.

Acknowledgment

The research was supported in part by a grant from National Basic Research Program (973 program) under grant No.2014CB347800, by a grant from National Natural Science Foundation of China (NSFC) under grant No.61370232. The corresponding author is Fangming Liu.

References

[1] Amazon elastic compute cloud. [Online]. Available: http://aws.amazon.com

[2] F. Xu, F. Liu, H. Jin, and A. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11–31, Jan 2014.

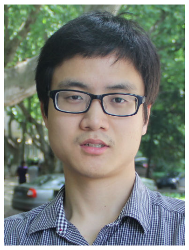
[3] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, and B. Li, "iaware: Making live migration of virtual machines interference-aware in the cloud," *Computers, IEEE Transactions on*, vol. 63, no. 12, pp. 3012–3025, Dec 2014.

[4] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: Sharing the network in cloud computing," in *ACM SIGCOMM*, 2012.

[5] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM*, 2011.

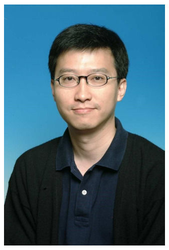
[6] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *USENIX NSDI*, 2011.

[7] J. Guo, F. Liu, H. Tang, Y. Lian, H. Jin, and J. Lui, “Falloc: Fair network bandwidth allocation in iaas datacenters via a bargaining game approach,” in *IEEE ICNP*, 2013.
[8] C. Guo, G. Lu, H. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, “Secondnet: A data center network virtualization architecture with bandwidth guarantees,” in *ACM CoNEXT*, 2010.
[9] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, “Improving datacenter performance and robustness with multipath tcp,” in *ACM SIGCOMM*, 2011.
[10] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, “Vl2: a scalable and flexible data center network,” in *ACM SIGCOMM*, 2009.
[11] J. Guo, F. Liu, D. Zeng, J. C. Lui, and H. Jin, “A cooperative game based allocation for sharing data center networks,” in *IEEE INFOCOM*, 2013.
[12] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: measurements & analysis,” in *ACM IMC*, 2009.
[13] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, “Elasticswitch: Practicalwork-conserving bandwidth guarantees for cloud computing,” in *ACM SIGCOMM*, 2013.
[14] R. Mazumdar, L. G. Mason, and C. Douligeris, “Fairness in network optimal flow control: Optimality of product forms,” *Communications, IEEE Transactions on*, vol. 39, no. 5, pp. 775–782, 1991.
[15] F. Kelly, “Charging and rate control for elastic traffic,” *European transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.
[16] H. Yaïche, R. Mazumdar, and C. Rosenberg, “A game theoretic framework for bandwidth allocation and pricing in broadband networks,” *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 5, 2000.
[17] F. P. Kelly, A. K. Maulloo, and D. K. Tan, “Rate control for communication networks: shadow prices, proportional fairness and stability,” *Journal of the Operational Research society*, pp. 237–252, 1998.
[18] H. Boche, M. Schubert, N. Vucic, and S. Naik, “Non-symmetric nash bargaining solution for resource allocation in wireless networks and connection to interference calculus,” in *Proc. 15th European Signal Processing Conference*, 2007.
[19] N. Nisan, *Algorithmic game theory*. Cambridge Univ Pr, 2007.
[20] S. Shakkottai and R. Srikant, *Network optimization and control*. Now Publishers Inc, 2008.
[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[22] Y. Chen, S. Alspaugh, and R. Katz, “Interactive analytical processing in big data systems: a cross-industry study of mapreduce workloads,” in *VLDB*, 2012.
[23] D. Xie, N. Ding, Y. Hu, and R. Kompella, “The only constant is change: incorporating time-varying network reservations in data centers,” in *ACM SIGCOMM*, 2012.
[24] J. Zhu, D. Li, J. Wu, H. Liu, Y. Zhang, and J. Zhang, “Towards bandwidth guarantee in multi-tenancy cloud computing networks,” in *IEEE ICNP*, 2012.
[25] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, C. Kim, and A. Greenberg, “Eyeq: practical network performance isolation at the edge,” in *USENIX NSDI*, 2013.
[26] J. Guo, F. Liu, X. Huang, J. C. Lui, M. Hu, Q. Gao, and H. Jin, “On efficient bandwidth allocation for traffic variability in datacenters,” in *IEEE INFOCOM*, 2014.
[27] T. Lam and G. Varghese, “Netshare: Virtualizing bandwidth within the cloud,” UCSD, Tech. Rep., 2009.



Jian Guo received his B.S. degree in School of Computer Science and Technology, Huazhong University of Science and Technology, China. He is currently a Ph.D. student in School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include data center networking and software-defined networking.

Fangming Liu is an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China; and he is awarded as the CHUTIAN Scholar of Hubei Province, China. He is the Youth Scientist of National 973 Basic Research Program Project on Software-defined Networking (SDN)-based Cloud Datacenter Networks, which is one of the largest SDN projects in China. Since 2012, he has also been invited as a StarTrack Visiting Young Faculty in Microsoft Research Asia (MSRA), Beijing. He received his B.Engr. degree in 2005 from Department of Computer Science and Technology, Tsinghua University, Beijing; and his Ph.D. degree in Computer Science and Engineering from the Hong Kong University of Science and Technology in 2011. From 2009 to 2010, he was a visiting scholar at the Department of Electrical and Computer Engineering, University of Toronto, Canada. He was the recipient of two Best Paper Awards from IEEE GLOBECOM 2011 and IEEE CloudCom 2012, respectively. His research interests include cloud computing and datacenter networking, mobile cloud, green computing and communications, software-defined networking and virtualization technology, large-scale Internet content distribution and video streaming systems. He is a member of IEEE and ACM, as well as a member of the China Computer Federation (CCF) Internet Technical Committee. He has been a Guest Editor for IEEE Network Magazine, an Associate Editor for Frontiers of Computer Science, and served as TPC for IEEE INFOCOM 2013-2015, ICNP 2014, ICDCS 2015, and ACM Multimedia 2014.



John C. S. Lui is currently a professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. He received his Ph.D. in Computer Science from UCLA. His current research interests are in communication networks, network/system security (e.g., cloud security, mobile security, etc.), network economics, network sciences (e.g., online social networks, information spreading, etc.), cloud computing, large scale distributed systems and performance evaluation theory. John serves in the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Performance Evaluation and International Journal of Network Security. John was the chairman of the CSE Department from 2005 to 2011. He received various departmental teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He is also a corecipient of the IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 Best Student Paper Awards. He is an elected member of the IFIP WG 7.3, Fellow of ACM, Fellow of IEEE and Croucher Senior Research Fellow.

Hai Jin is a Cheung Kung Scholars Chair Professor of computer science and engineering at the Huazhong University of Science and Technology (HUST), China. He is now dean of the School of Computer Science and Technology at HUST. He received his Ph.D. degree in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked at the University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded the Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and chief scientist of the National 973 Basic Research Program Project of Virtualization Technology of Computing Systems. He is a senior member of the IEEE and a member of the ACM. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security.