Falloc: Fair Network Bandwidth Allocation in IaaS Datacenters via a Bargaining Game Approach

Jian Guo[1] Fangming Liu*[1] Haowen Tang[1] Yingnan Lian[1] Hai Jin[1] John C.S. Lui[2]
[1]Key Laboratory of Services Computing Technology and System, Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology, China.
[2]Department of Computer Science & Engineering, The Chinese University of Hong Kong.
[1]{guojian,fmliu,cstang02,hjin}@hust.edu.cn, [2]cslui@cse.cuhk.edu.hk

***Abstract*—With wide application of virtualization technology, tenants are able to access isolated cloud services by renting the shared resources in 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 network isolation. In this paper, we propose *Falloc*, a new bandwidth allocation protocol, 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 *weights* of VMs. To design *Falloc*, we model the datacenter bandwidth allocation as a bargaining game and propose a distributed algorithm to achieve the asymmetric Nash bargaining solution (NBS). We apply the theory to practice by implementing *Falloc* with OpenFlow in experiments under diversed scenarios, which shows that *Falloc* can achieve fairness by adapting to different network requirements of VMs, and balance the tradeoff between bandwidth guarantee and proportional bandwidth share. By carrying out large scale trace-driven simulations using real-world Mapreduce workload, we show that *Falloc* achieves high utilization and maintains fairness among VMs in datacenters.**

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 set of virtual machines (VMs) with performance isolation on CPU and memory resources. However, in current datacenters, the scarce network bandwidth is shared across many tenants without any performance guarantees. 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 on attracting more tenants. As the rapid development in infrastructure 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 protect each application's performance in the competition of various network aggressive applications (e.g., [2]–[4]). That is, a fair policy should be able to allocate bandwidth according to the network requirement of applications. Current cloud applications have two primary requirements for datacenter networks, 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.

However, the tradeoff between guaranteeing bandwidth and sharing bandwidth proportionally [5], [6] makes it a challenging work to fulfill these two basic requirements in datacenter networks. Even more serious is that the allocation should achieve high network utilization. Recent works based on bandwidth competition, such as Seawall [7], proportionally divide the network to VMs or tenants according to their weights, by introducing flow-level fairness to the VM-level. They neglect the requirement of minimal bandwidth guarantee since the sharing bandwidth of a VM can reduce significantly if more VMs compete for the same physical bandwidth resources. Other works based on VM placement such as Oktpus [8] provide bandwidth guarantees for VMs by assigning them to locations where sufficient bandwidth can be reserved. Due to the time-varying nature of the network traffic in datacenters, the reservation policy leads to bandwidth wastage if the reserved bandwidth are not fully utilized.

Although these two mechanisms emphasize on different goals and both have their own merits and demerits, they can potentially be combined to exploit both of their advantages. Specifically, by assigning VMs to proper locations, one could achieve a specified bandwidth guarantees (fixed or time-varying) for each VM. When the network demand deviates

*The Corresponding Author is Fangming Liu. The research was supported in part by a grant from National Basic Research Program (973 program) under Grant of 2014CB347800, by a grant from National Natural Science Foundation of China under grant No.61133006.
978-1-4799-1270-4/13/$31.00

away from this pre-reserved bandwidth, one could deploy competition mechanism to allocate the network according to the bandwidth demand, which will improve the network utilization. The key challenge is how much bandwidth should be allocated to each VM. While existing works focus on 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. By taking advantage of a game-theoretical approach, this paper takes the first step to model the bandwidth allocation process in datacenters as a *asymmetric weighted Nash bargaining game*, where all VMs are cooperative so as to maximize the social welfare in a manner without harming others' benefit.

Contribution: We make the following contributions in this work: *First*, based on our derived theoretical guidelines, we present the design of *Falloc*, an application layer bandwidth allocation protocol to achieve fairness among VMs in datacenters. In *Falloc*, each VM is assigned with a *base bandwidth* and a *weight*. The protocol 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. Based on a bargaining game approach, the calculation of the bandwidth allocation to each VM-pair can be executed in a distributed manner and the result achieves a weighted Nash bargaining solution. *Second*, we implement the *Falloc* prototype with OpenFlow and evaluate it on a Mininet testbed under diversed scenarios. By characterizing the impact of both B and K on the allocation, we validates *Falloc*'s ability to enforce bandwidth guarantee and proportional share on network bandwidth, and show that *Falloc* can balance the tradeoff between them by changing the base bandwidth of VMs. *Third*, 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.

II. Motivation and Design Objectives

A. Motivation and Objectives: Fair Datacenter Networks

The fairness concepts in the context of datacenter are introduced from traditional network resource sharing problems. The most remarkable difference between these two scenarios is that, conventional transport layer fairness equally slice the bandwidth among all flows, while the fundamental of bandwidth sharing in datacenter network is to achieve predicable/high performance for applications [8]. As applications are running by VMs in IaaS datacenters, researchers choose to reserve certain bandwidth for different VMs (e.g., [8]), or allocate a certain portion of the bandwidth on congested links to VMs (e.g., [7]) based on the bandwidth requirement of these applications. These previous works indicate that a network sharing policy for IaaS datacenters should try to guarantee fairness by providing network isolation for different VMs. Specifically, cloud providers need to satisfy the bandwidth demand of one VM without sacrificing others' network performance. However, it is not easy to achieve fairness relying on traditonal 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 TPC-based applications in the cloud.

Hence, to fairly share the intra-datacenter networks, we need to design an *application-layer bandwidth sharing protocol* with the ability to fulfill the bandwidth requirement of VMs running different applications. We believe the following important objectives should be taken into consideration:

• **Bandwidth guarantee.** With bandwidth guarantees for VMs, one can achieve predicable performance for network sensitive applications running in these VMs. For many cloud applications, predicable performance means less delay for application users. For example, a web service can provide fast data delivery to users if the data transfer between the server's front and back end is guaranteed [9]. With this protocol, cloud users can specify bandwidth for each VM and the protocol should guarantee this bandwidth for the VMs.

• **Weight assignment.** Since jobs in the cloud have different priorities, the protocol 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, a cloud provider may want to allocate more bandwidth for the important job. With the weights of VMs, the protocol should share the bandwidth in proportion to their respective weights.

However, these two objectives both have their disadvantages: bandwidth guarantee may cause a wastage if the guaranteed bandwidth is left unused due to the time-varying network requirements of cloud applications, and proportional bandwidth share can not provide a reliable network performance for applications since the shared bandwidth reduces as more competitors take part in. 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 requirement. Since proportional bandwidth share can highly utilize the bandwidth on congested link, we can prioritize bandwidth guarantee while judging how much to guarantee, and share the residual bandwidth in proportion to the weight of each VM. This way, we can also achieve a high utilization of the network bandwidth.

B. Our Choice: Base Bandwidth and Weight for VMs

To achieve our goal of fairness in sharing datacenter networks, we introduce the definition of *base bandwidth* (B) and *weight* (K). The base bandwidth of each VM is a threshold for 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 exceeded the base bandwidth is shared in proportion to the weight. 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. The unpredictability in network performance leads to uncertainty in execution times for network intensive jobs, which indirectly translates into a

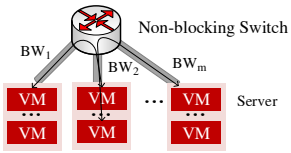


Fig. 1. Datacenter model with m servers connecting to a non-blocking switch with heterogeneous bandwidth. The VMs are hosted on these servers.

risk of revenue loss for the cloud users. 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., [8], [10]) with our competition based mechanisms. An allocation based mechanism studies how to place VMs to the servers with bandwidth reservation, and 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 the bandwidth demand. For example, Oktpus [8] 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 protocol to allocate the unused bandwidth to VMs with bandwidth demand higher than the base bandwidth.

• **Flexible fairness.** Due to the limited network resource 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 applications, and uses 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 while attempt to share the residual bandwidth in proportion to VMs' weights.

III. Resource Sharing Problem in Datacenter Network

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

A. Modeling Datacenter Network

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. Since existing work such as multi-path routing (e.g., [11]) and multi-tree topologies (e.g., [12]) has largely improved bisection bandwidth of datacenter networks, we design our policy with a full bisection bandwidth network, where the physical bandwidth of servers can be fully utilized without considering the bottlenecks inside the datacenter networks. Based on the hose model in [5], [8], we present the datacenter network abstraction in Fig. 1, where the servers are connected to a non-blocking virtual switch.

In our model, we propose a bandwidth allocation policy for each VM-pair, which provides strong guarantees compared to the solution of allocating bandwidth to each VM (e.g., [8]).

TABLE I. NOTATIONS

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

The VM-pair based allocation policy offers better resource management than a VM based policy since some applications need network performance guarantee between VM-pairs. For example, a reduce task of a MapReduce job needs to shuffle data from multiple map tasks, and the job may be delayed by one or more slow tasks on network congested VMs. Moreover, by specifying bandwidth of each VM-pair, the resulting bandwidth of each VM can be easily obtained.

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 rates) 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 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 full-duplex Ethernet adapter, then we have $C_m = C_m^I = C_m^E$. In order to guarantee the base bandwidth in extreme cases that all the traffic demands of VMs are aggressive, the sum of base bandwidth of all the VMs hosted on the same server should be 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 protocol design, and evaluate the performance in the simulations.

B. Constraints Formulation

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. In order to meet the network requirement when the bandwidth demand of a VM is less than the baseband width, the allocation should ensure an initial lower bound for each VM-pair assocaited with this VM, i.e., $r_{i,j} \geq L_{i,j}$ where

$$L_{i,j} = \min\{D_{i,j}, B_{i,j}\}. \quad (1)$$

$B_{i,j}$ represents the base bandwidth for the VM-pair from VM i to VM j, which is unknown and can be derived from

the base bandwidth associated with VM i and VM j. In our work, we specify $B_{i,j}$ as the following form

$$B_{i,j} = \min\{B_i^E \frac{K_i}{\sum_{D_{ik}\neq 0, k\in\mathcal{N}} K_k}, B_j^I \frac{K_j}{\sum_{D_{kj}\neq 0, k\in\mathcal{N}} K_k}\}, \tag{2}$$

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

$$K_{i,j} = \frac{K_i}{\sum_{D_{ik}\neq 0, k\in\mathcal{N}} 1} + \frac{K_j}{\sum_{D_{kj}\neq 0, k\in\mathcal{N}} 1}. \tag{3}$$

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. Due to page limit, we put the details in our technical report [13].

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

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

This ensures that the allocated bandwidth can be fully utilized.

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 VM-pairs 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 VM-pairs whose bandwidth demand is higher than the base bandwidth, the allocation policy should not only allocate each VM-pair 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-pairs without sacrificing the bandwidth of other VM-pairs.

C. Asymmetric Nash Bargaining Solution (NBS) for Fairness

With consideration on both utilization and fairness, we choose to use the Nash bargaining solution 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 as the bandwidth allocation in datacenters, where each VM-pair should be guaranteed with the initial base 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 the NBS is a suitable alternative for our allocation policy in the context of datacenter networks.

To formulate the optimization problem of bandwidth allocation, we first present the main concepts and results 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 weight, we apply the asymmetric weighted Nash bargaining solution [14] and assign them with different contributions to the social welfare by using the exponentiation of the utility gains, i.e., $(r_{i,j} - L_{i,j})^{K_{i,j}}$.

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. Since 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 [15].

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 bandwidth compared to their initial rates. 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 $\max \sum \ln(r_{i,j} - L_{i,j}), \forall r_{i,j} \in J$.

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

$$\max_{r_{i,j}} \quad \sum_j \sum_i K_{i,j}\ \ln(r_{i,j} - L_{i,j}), \forall r_{i,j} \in J \tag{6}$$

$$\text{s.t.} \quad L_{i,j} \leq r_{i,j} \leq U_{i,j},\ \forall i, j \in \mathcal{N} \tag{7}$$

$$\sum_{i\in V_m} r_i^I \leq C_m\ \forall m \in \mathcal{M} \tag{8}$$

$$\sum_{i\in V_m} r_i^E \leq C_m,\ \forall m \in \mathcal{M}, \tag{9}$$

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. (8) and Eq. (9) represent the constraints of the bandwidth capacity for each server.

Note that the VM-pairs in the objective always has a rate $r_{i,j} > L_{i,j}$ and the rates of other VM-pairs are equal to their lower bounds as discussed in Sec.III-B. For simplicity, we have $r_{i,j} > L_{i,j}, \forall i, j \in \mathcal{N}$ when characterizing the optimal solution. In fact, it turns out that the rates satisfying $r_{i,j} = L_{i,j}$ can be eliminated in the differential in Sec.IV-A and this assumption has no impact on the results.

IV. Protocol Design via Bargaining Game

In this section, a bargaining game approach is used to construct iterations which convergence to the multipliers that solve the optimal rate for each VM-pair. We present our *Falloc* protocol design 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 the 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: 1 means i is on server m, and 0 otherwise. 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 [17] 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 $i, j \in \mathcal{N}$:

$$r_{i,j}^* = L_{i,j} + \frac{K_{i,j}}{\sum_{m=1}^M \lambda_m^E w_{m,i} + \sum_{m=1}^M \lambda_m^I w_{m,j}}, \quad L_{i,j} \leq r_{i,j} \leq U_{i,j}, \tag{10}$$

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

The proof of Theorem 2 is presented in the appendix of the technical report [13].

The original problem in Eq. (6)-(9) 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. (10) 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 from other servers. This motivates us to obtain the rate of each VM-pair *distributively* rather using a centralized approach for the optimization.

B. Algorithm via Bargaining Game

The centralized primal problem in Eq. (6)-(9) 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 dual problem (P_d) is described as follows:

$$\max_{\lambda^I, \lambda^E \in \mathcal{R}^M} d(\lambda^I, \lambda^E) = \mathcal{L}(r^*, \lambda^I, \lambda^E), \tag{11}$$

where $d(\lambda^I, \lambda^E)$ is the dual function and $\mathcal{L}(\cdot)$ is the Lagrangian of the primal problem. The derivation of the dual problem is presented in our technical report [13].

To solve the primal problem, we first obtain the optimal solution to the dual problem. Specifically, by using a suitable step-size, we design an iteration that converges to the optimal λ^I (λ^E) by applying the subgradient methods [18]. Since the strong duality holds as discussed above, we can achieve the optimal Nash bargaining solution with Eq. (10).

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:

$$\lambda_m^{(s+1)} = \max(0, \lambda_m^{(s)} + \xi \frac{\partial d}{\partial \lambda_m}), \forall m \in \mathcal{M}, \tag{12}$$

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

It has been proved in [18] 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* [18], then the recursion of dual variable $\{(\lambda^I)^{(s)}\}$ converges, thus

$$\lim_{s\to\infty} (\lambda^I)^{(s)} = \lambda^{I*} \in \overline{\lambda}. \tag{13}$$

Having determined the step size in Eq. (12), the gradient of the dual function can also be solved by obtaining the partial derivatives of $d(\lambda^I, \lambda^E)$ as below

$$\frac{\partial d(\lambda^I, \lambda^E)}{\partial \lambda_m^I} = \sum_{j=1}^N w_{m,j} \sum_{i=1}^N r_{i,j}^* - C_m. \tag{14}$$

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

$$\frac{\partial d(\lambda^I, \lambda^E)}{\partial \lambda_m^E} = \sum_{i=1}^N w_{m,i} \sum_{j=1}^N r_{i,j}^* - C_m. \tag{15}$$

In Theorem 2, we have obtained the explicit form of optimal rate $r_{i,j}^*$. The sequences generated by Eq. (12) converge to the optimal solution to the dual problem (P_d) in Eq. (11) 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 $\forall i, j \in \mathcal{N}$

$$\lim_{s\to\infty} r((\lambda^I)^{(s)}, (\lambda^E)^{(s)}) = r^* \in \overline{R}. \tag{16}$$

In summary, the approach to obtain the optimal rate can be viewed as an iterative bargaining process, where the dual variables serve as the bargaining prices (Eq. (12)) and the rates indicate the utility gains of the players (Eq. (10)). For example, $r_{i,j}$ is the rate of VM-pair i and 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_m^E and λ_l^I, respectively. 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. If server m has any remaining bandwidth, i.e., $C_m > r_{p_m}^E$, it cuts down the price of the bandwidth. Then the VM-pair will buy more bandwidth in the next round when indicated by the increase of rate $r_{i,j}$ as the dual variables decreases. Therefore, the remaining bandwidth can be allocated to the VM-pairs with unsatisfied demand.

Similarly, when the total allocated rates exceed the capacity of server m, the price will increase so as to reduce the excessively allocated bandwidth

C. Falloc: Protocol Design

According to the results in the bargaining game approach, the process of obtaining the optimal rate can be executed in a distributed manner: 1) the dual variables (λ_m^E and λ_m^I) of each server m can be updated independently with local information in Eq. (15). 2) the iteration of each rate ($r_{i,j}$) in Eq. (10) only requires the information of the server hosting VM i and the server hosting VM j. This motivates the design of *Falloc*, which consists of following mechanisms.

1) Centralized convergence control. To design *Falloc*, we first consider how to coordinate all the iteration processes and where to locate the base bandwidth and weight of each VM. Instead of a global controller, it is much more practical to use several centralized servers in different local areas to control the convergence and manage the global information of VMs. This is because in IaaS datacenters, the intra-datacenter traffic is always limited within the VMs belonging to the same tenant, and a reasonable VM allocation strategy should try to place the VMs of the same tenant into the same rack or aggregation. For example, in a tree-topology datacenter, we can use only one centralized server under each aggregate or top-of-rack switch. This can avoid the messages across the core switch, and greatly reduce the convergence steps of our algorithm due to the decrease in the number of VM-pairs.

As shown in Algorithm 1, the centralized controller first initializes the lower/upper bound bandwidth for each VM-pair (line 1-3), and then uses the stopping criteria (line 6) to control the convergence of Algorithm 2. In the convergence process, a precise result will lead to heavy cost since it may cause remarkable increase in iteration steps. We can define two stopping rules in *Falloc* protocol to balance the tradeoff between cost and precision.

- **Step-based mechanism:** Use S as the total iteration steps in the convergence process. 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., $\Delta = 0.1\,\text{Mbps}$) so as to fulfill his agreement.

Before the iteration, the required input data including the stopping rules is sent out by the controller. The iteration servers then locally update the iterations until every $r_{i,j}$ converges.

2) 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. The iteration of $r_{i,j}$ starts with the initial lower bound $L_{i,j}$. We present the iteration process of each server in Algorithm 2.

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. (12) (line 2). For all the VM-pairs ($r_{i,j}$) with VM i on server m, the server should request for 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. (10). In addition, we make a judgment in case that the rate exceeds the upper bound bandwidth $U_{i,j}$ (line 6). The step size is chosen according to the step size rules in Proposition 1 and updated locally within each iteration (line 12).

Algorithm 1 *Falloc*: Centralized convergence control

Input:
The base bandwidth of VMs $< B_i^I, B_i^E >, \forall i \in \mathcal{N}$
The weight of VMs $< K_i >, \forall i \in \mathcal{N}$
The total number of iteration rounds: S
The gap between two consecutive iterations: Δ
Output: Rate allocation matrix: $[r_{i,j}]_{N\times N}, \forall i \in \mathcal{N}$
for all VM-pair: $i, j \in \mathcal{N}$ **do**
Initialize $L_{i,j}$ as Eq. (1) and $U_{i,j}$ as Eq. (4)
end for
for all Server: $m \in \mathcal{M}$ **do**
$r_{i,j}^{(0)} = L_{i,j}$
while $s < S$ or $r_{i,j}^{(s)} - r_{i,j}^{(s-1)} > \Delta$ **do**
Run Algorith 2 at server m
end while
end for

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 an approximate optimal value and then slowly approach the optimal one. Hence, the algorithm can finish within an acceptable number of steps if we do not need a strict optimal rates for all VM-pairs.

When the iteration stops under the control of the stopping rules, *Falloc*'s outputs can be applied in hypervisors or swtiches 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.

3) 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. 2 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 for a dual variable (λ_l^I), it first checks if it has received the dual variable. If it has the dual variable (Fig. 2(b)), it will continue the iteration with this value. Otherwise (Fig. 2(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 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 re-sends an ACK consisting of the dual variable until it finishes the calculation.

On the other hand, when a *reply* server finishes calculating

Algorithm 2 *Falloc*: Iteration process on server m

Input:
The step-size: ξ
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: Rates of VM-pairs on server m: $r_{i,j}, i \in V_m$

Update allocated bandwidth
$r^E_{p_m} = \sum_i w_{m,i} r^E_i$, $r^I_{p_m} = \sum_i w_{m,i} r^I_i$
Update dual variables as Eq. (12)
$\lambda^E_m = \max(0, \lambda^E_m - \xi(C_m - r^E_{p_m}))$
$\lambda^I_m = \max(0, \lambda^I_m - \xi(C_m - r^I_{p_m}))$
for all $r_{i,j}, i \in V_m$ **do**
  Update $\lambda^E = \lambda^E_m$
  Obtain $\lambda^I = \lambda^I_l$ from server l, $j \in V_l$
  if $\frac{K_{i,j}}{\lambda^E+\lambda^I} > U_{i,,j} - L_{i,j}$ **then**
    $r^{(s)}_{i,j} = U_{i,j}$
  else
    $r^{(s)}_{i,j} = L_{i,j} + \frac{K_{i,j}}{\lambda^E+\lambda^I}$
  end if
end for
Update step size ξ
Update iteration round $s = s + 1$

a new dual variable, it will first check if there are REQUESTs for this variable. If there are (Fig. 2(a)), it will send ACKs with this dual variable to the request servers. Otherwise (Fig. 2(b)), it pushes the dual variable to all the paired servers for future usage.

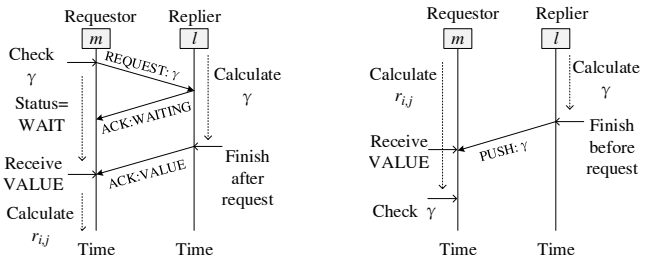
To implement *Falloc* in datacenters, we present two communication schemes for message exchange between controllers and iteration servers.

• **Distributed solution:** For traditional datacenters, the iteration process can be performed in a distributed manner on each server. When the iteration server receives a START command, it will request for the input data (as shown in Algorithm 2) from the controller. Usually, the information can be found on this controller. If not, the controller will broadcast the required information to all the centralized controllers in other areas and then feed it back to the iteration servers. 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.

• **Centralized solution:** In a datacenter deploying Software-Defined Networking (SDN), we can perform the iteration process in a centralized manner. Under such condition, the iteration process at each server can be executed by a single process in the controller's operating system. Hence, the delay and overhead of transmitting the dual variables and input data can be avoided. After obtaining the optimal rates, the controller will allocate the bandwidth for each VM-pair using the SDN switch. This distributed solution only has M message exchange, but requires $S \cdot O(n^2)$ computational complexity on the centralized controller.

V. Implementation with OpenFlow and Evaluation

In this paper, we implement *Falloc* with the OpenFlow protocol by using the centralized solution as described in Sec. IV-C. (Note that *Falloc* can also be implemented distributedly at the source) OpenFlow is an example of SDN, where the



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

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

Fig. 2. Message exchange protocol between request server and reply server.

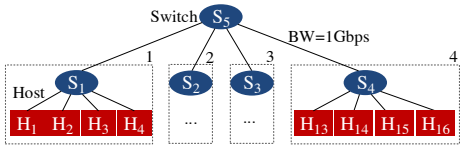


Fig. 3. Testbed with 16 hosts divided into 4 groups. Each group consists of 4 hosts and 1 switch. The group is equivalent to a server.

path of network packets through the switches can be determined by software running in switches or servers. The *Falloc* prototype runs our proposed bandwidth allocation algorithm in a centralized server, and enforces the allocation result by forwarding packets through specified queues in the switches. It consists of two basic components:

• **Bandwidth allocation service:** The service is responsible for calculating the rates to be allocated, and enforce rate allocation for each VM-pair. It reads the input data following the message exchange protocols, and then performs the algorithms in Sec. IV-C to obtain the optimal rate for each VM-pair. After finishing the calculation, it will set up queues with maximum rates at the port of each switch, based on the allocated rate of the VM connected to this port.

• **OpenFlow controller:** The controller is a program running on the centralized server, which manages packet-forwarding for VMs in the network. We implement it as a L3-learning switch with basic functions such as Address Resolution Protocol (ARP), flow table configuration for IPv4 packets, etc. To realize rate control in *Falloc*, the controller forwards a packet to the specific queue by configuring the flow table in OpenFlow switch, based on the packet's source and destination.

A. Experimental Results with OpenFlow

We evaluate *Falloc* prototype in Mininet, a SDN evaluation platform running real network protocols and workloads, with which the developed code can be moved to a real OpenFlow network without any changes. 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. 3, 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 1 Gbps 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 and the observed rates of H_1, H_2, H_3 and H_4 under typical scenarios.

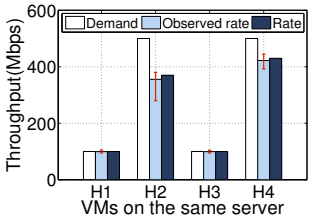


Fig. 4. Bandwidth allocation to VMs with different bandwidth demand, while the base bandwidth of each VM is 250 Mbps.

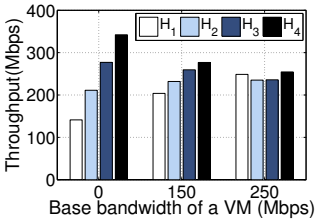


Fig. 5. Bandwidth allocation to VMs by varying the base bandwidth, when $K_1 : K_2 : K_3 : K_4 = 1 : 2 : 3 : 4$.

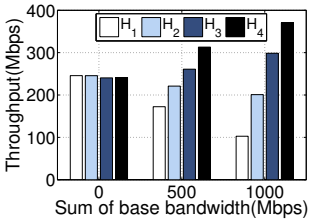


Fig. 6. Bandwidth allocation to VMs with different base bandwidth for each VM.

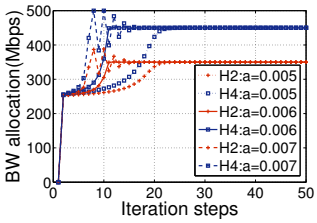


Fig. 7. Rate variation of VM H_2 and H_4 with increasing number of iteration steps.

Results of bandwidth allocation. Fig. 4 shows the bandwidth allocation to the VMs on server 1, where each VM has a base bandwidth of 250 Mbps and all the VMs have the same weight. H_1 and H_2 are pulling data from server 2 and each is connected to one VM, while H_3 and H_4 are pulling data from server 3 and 4, respectively, and each is connected to two VMs. For VM H_1 and H_3, whose bandwidth demand is less than the base bandwidth, *Falloc* can guarantee sufficient bandwidth for both of them. This implies that the base bandwidth can be guaranteed in our implementation. For VM H_2 and H_4, whose bandwidth demand exceeds the base bandwidth, *Falloc* gives a fair bandwidth allocation, by sharing the residual bandwidth with about a proportion of $2 : 3$, which is exactly the proportion of the total weight associated with H_2 to H_4. The result also indicates that our implementation can achieve 99% accuracy when controlling the rate with 100 Mbps network workload and above 95% accuracy when the workload increases to about 400 Mbps.

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. 5 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. 5 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 remaining bandwidth to be shared after guaranteeing the bandwidth demand for each VM. Note that the difference of bandwidth allocation 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 the VMs. This is showed by the proportion of the rates with $B = 0$ in the figure, which is about $2 : 3 : 4 : 5$. Considering the weights of VMs connecting to $H_1 \sim H_4$, the allocation result is reasonable.

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. 6, 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 maximumed, i.e., equals the 1000 Mbps physical bandwidth, the proportion of bandwidth for $H_1 : H_2 : H_3 : H_4$ is strictly $1 : 2 : 3 : 4$, although they are assigned with the same weight. This is because the *Falloc* protocol should hardly guarantee the bandwidth demand if it is less than the base bandwidth.

Balance the tradeoff. The above analysis for Fig. 5 and Fig. 6 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. 4. 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. 7, 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. 8 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 converges 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 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.

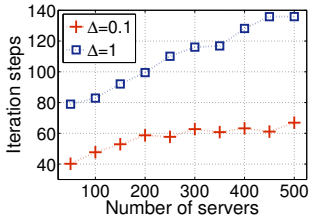


Fig. 8. Average convergence steps with increasing number of servers when $\Delta = 1$ and 0.1 Mbps.

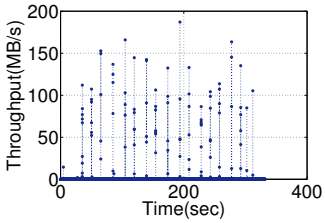


Fig. 9. Network throughput of VM when running Hadoop Word Count with 1.2 GB input data.

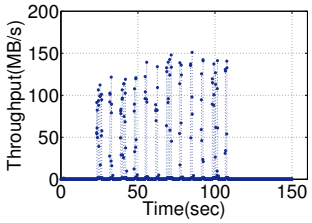


Fig. 10. Network throughput of VM when running Hadoop Sort with 1.0 GB input data.

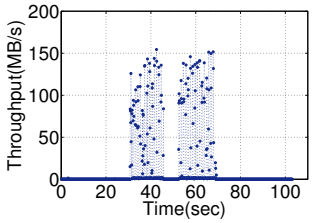


Fig. 11. Network throughput of VM when running Hadoop Join with 1.0 GB input data.

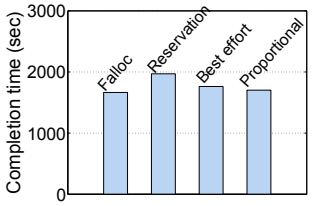


Fig. 12. Comparing the completion time between *Falloc* and other sharing policies.

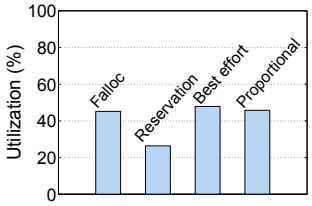


Fig. 13. Comparing the average network utilization between *Falloc* and other sharing policies.

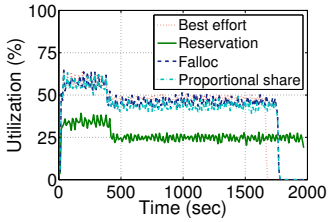


Fig. 14. Comparing network utilization of the entire datacenter during the execution of all batched jobs.

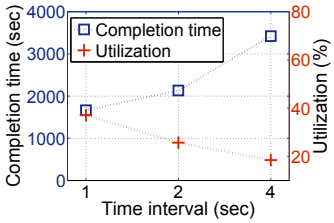


Fig. 15. Job completion time and average network utilization with different time $\Delta t = 1, 2, 4$ s.

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 cpu and 32 GB memory, running KVM based virtual machines, and each VM is allocated with one core and 4 GB memory. Fig. 9-11 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 [19], we can use these capped rates as the 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 [5]. 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 [8]). 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. To compare the job completion time, the simulator runs the same batched jobs with each sharing policy at a time. The input iteration steps of *Falloc* is set to 50 and it 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. But since the workloads in the simulation have similar bandwidth demand, *Falloc* shows no advantage on providing fairness for different jobs, comparing to best effort and proportional sharing policies. We leave the comparison consisting of aggressive network application (e.g., video delivery) as future work.

Network utilization. To understand how *Falloc* can significantly reduce 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. 14), 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 cost. We repeat the above simulations by varying the main factors impacting the cost of running *Falloc*, i.e., the updating time interval Δt. As shown in Fig. 15, the completion time increases as the Δt varies from 1 to 4 s while the network utilization decreases. 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. Hence, the improvements in performance can be achieved by increasing the precision of *Falloc*'s output, which will lead to more overhead in the server and network. In order to reduce the overhead, a practical method is to reduce the iteration rounds, since the iteration will become useless after the algorithm converges.

VI. 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 and the proposed mechanisms consists of two main basic ideas, i.e., guaranteeing bandwidth for VMs and proportionally sharing bandwidth among VMs or tenants [6].

Previous work, such as Oktopus [8], SeconNet [10] and [20], focuses on providing deterministic bandwidth guarantees for VMs. They allocate VMs into servers based on VMs' bandwidth requirements, and by enforcing 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 variation in bandwidth demand of cloud applications. In [19], the authors propose a time varying reservation policy to meet the bandwidth requirements of VMs, and make a contribution to increase the utilization of the datacenter when reserving bandwidth for VMs. However, it is only suitable for pulse-like bandwidth demand. Unlike the VM placement based methods above, Gatekeeper [21] provides minimum bandwidth guarantee for VMs by shaping the traffic of VMs, but the unused bandwidth is shared in a best-effort manner.

Other works provide network isolation for VMs by sharing the bandwidth resources proportionally. Seawall [7] provides a hypivisor based mechanism to slice the bandwidth of each congested link according to weights of source VMs. NetShare [22] allocates the relative bandwidth among different services based on their weights to and provides constant proportionality of tenants throughout the network. Faircloud [5] presents understanding on key requirements and properties for sharing network in datacenters. The authors propose three bandwidth allocation policies based on proportional share to explore the tradeoff in sharing datacenter networks. These weight based competition mechanisms, however, can not provide deterministic bandwidth guarantees for VMs since the competition of other VMs should be considered in bandwidth allocation.

Finally, similar to [14], [16], [23], [24], our work use the cooperative game framework to share resources.

VII. Conclusion

In summary, we propose a bandwidth allocation protocol named *Falloc* to fairly share network resources at VM-level in IaaS datacenters. *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. Through a bargaining game approach, we present the design of *Falloc* protocol for both distributed and centralized environment in a datacenter. The experiment with OpenFlow implementation shows that *Falloc* can provide flexible fairness for VMs by balancing the tradeoff between bandwidth guarantee and proportional bandwidth share. We carry out trace-driven simulations to evaluate the performance of *Falloc*, and show that the protocol can achieve high network utilization and good job completion time in datacenter network.

References

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

[2] F. Liu, Y. Sun, B. Li, B. Li, and X. Zhang, "FS2You: Peer-Assisted Semi-Persistent Online Hosting at a Large Scale," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 10, pp. 1442–1457, 2010.

[3] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing Resource-Poor Mobile Devices with Powerful Clouds: Architectures, Challenges and Applications," *IEEE Wireless Communications Magazine, Special Issue on Mobile Cloud Computing*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[4] F. Liu, B. Li, L. Zhong, B. Li, H. Jin, and X. Liao, "Flash Crowd in P2P Live Streaming Systems: Fundamental Characteristics and Design Implications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 7, pp. 1227–1239, Jul. 2012.

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

[6] 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.

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

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

[9] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. C. Lui, and H. Jin, "Carbon-aware load balancing for geo-distributed cloud services," in *IEEE MASCOTS*, 2013.

[10] 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.

[11] 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.

[12] 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.

[13] J. Guo, F. Liu, H. Tang, Y. Lian, H. Jin, and L. John C.S., "Falloc: Fair network bandwidth allocation in iaas datacenters via a bargaining game approach," Technical report, HUST, http://grid.hust.edu.cn/fmliu/icnp2013falloc.pdf, August 2013.

[14] 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.

[15] N. Nisan, *Algorithmic game theory*. Cambridge Univ Pr, 2007.

[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] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[18] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, 2003.

[19] 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.

[20] 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.

[21] H. Rodrigues, J. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *USENIX Workshop on I/O Virtualization*, 2011.

[22] T. Lam and G. Varghese, "Netshare: Virtualizing bandwidth within the cloud," UCSD, Tech. Rep., 2009.

[23] Z. Fang and B. Bensaou, "Fair bandwidth sharing algorithms based on game theory frameworks for wireless ad-hoc networks," in *IEEE INFOCOM*, 2004.

[24] J. Dai, F. Liu, B. Li, B. Li, and J. Liu, "Collaborative Caching in Wireless Video Streaming Through Resource Auctions," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 2, pp. 458–466, Feb. 2012.