eBA: Efficient Bandwidth Guarantee Under Traffic Variability in Datacenters

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

***Abstract*— Datacenter networks suffer unpredictable performance due to a lack of application level bandwidth guarantees. A lot of attention has been drawn to solve this problem such as how to provide bandwidth guarantees for virtualized machines (VMs), proportional bandwidth share among tenants, and high network utilization under peak traffic. However, existing solutions fail to cope with highly dynamic traffic in datacenter networks. In this paper, we propose *eBA*, an efficient solution to bandwidth allocation that provides end-to-end bandwidth guarantee for VMs under large numbers of short flows and massive bursty traffic in datacenters. *eBA* leverages a novel distributed VM-to-VM rate control algorithm based on the logistic model under the control-theoretic framework. *eBA*'s implementation requires no changes to hardware or applications and can be deployed in standard protocol stack. The theoretical analysis and the experimental results show that *eBA* not only guarantees the bandwidth for VMs, but also provides fast convergence to efficiency and fairness, as well as smooth response to bursty traffic.**

***Index Terms*— IaaS datacenter network, traffic variability, bandwidth allocation, rate control, efficiency and fairness.**

I. Introduction

PUBLIC cloud (e.g., [1]) has been increasingly popular since it provides economical resources for deploying today's business in a wide area. Using a simple pay-as-you-go charging model, cloud providers are able to lease the resources to different cloud tenants in the form of Virtualized Machines (VMs), with isolated performance on CPU and memory. However, to the best of our knowledge, current datacenters do not offer bandwidth guarantee for tenants. The network performance between two VMs can fluctuate significantly due to the interference of network intensive applications on other VMs [2]–[4]. This unpredictable performance leads to uncertainty in execution times of jobs, which increases the risk of revenue loss to tenants. As a result, providing performance guarantee for intra-datacenter communication has received significant interest in the networking research community. There is a general consensus among researchers about the need for *basic bandwidth allocation requirements* [5], such as bandwidth guarantee for VMs, proportionally sharing bandwidth resource among tenants and a high utilization of network resources.

However, previous measurement works (e.g., [6], [7]) reveal the characteristics of datacenter traffic. First, there are large numbers of short flows in datacenter networks, among which about 80% of flows are less than 10KB in size and the average inter-arrival time can be as low as about 15ms at Top-of-Rack switches [7]. Second, short congestion periods are common across many links. Up to 86% of links observe short-lived congestions during one day [6]. Third, the datacenter traffic has significant variability. The average change in traffic over 10s is about 82% [6]. These unique characteristics make datacenter traffic remarkably different from any other network traffic. Hence, a practical bandwidth allocation should not only achieve the basic requirements, but also dynamically adapt to the traffic patterns in datacenters. Specifically, we consider the following *dynamic requirements*:

- *Handling of large numbers of short flows:* Short flows create bursty traffic in the network, which causes interferences to the throughput of long-lived TCP flows. As short flows are frequently generated, such traffic bursts may make other flows unstable and reduce the link utilization. The allocation policy should try to avoid sharp changes in throughput even at the presence of these short flows.
- *Smooth adaption in short congestion periods:* The allocation should change smoothly when congestions occur. If the rate-limit decreases the flows' bandwidth too sharply during congestions, the underlying TCP flows will fluctuate and the network performance will drop.
- *Fast convergence on bandwidth allocation:* The convergence of allocation process should be as fast as possible to keep pace with the significant variabilities in traffic. Otherwise, the allocation process cannot converge to the result that satisfies the basic requirements.

However, designing a bandwidth allocation algorithm to handle the dynamic nature of datacenter traffic is challenging. Fig. 1 shows the three key tasks to achieve efficient bandwidth

Manuscript received April 22, 2015; revised March 17, 2016; accepted June 26, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Kuri. Date of publication August 16, 2016; date of current version February 14, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61520106005 and in part by the National 973 Basic Research Program under Grant 2014CB347800. (*Corresponding author: Fangming Liu.*)

F. Liu and J. Guo are with the Services Computing Technology and System Laboratory and the Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: fmliu@hust.edu.cn; guojian@hust.edu.cn).

X. Huang is with the Ministry of Education Key Laboratory for Earth System Modeling, Center for Earth System Science, Tsinghua University, Beijing 100084, China (e-mail: hxm@tsinghua.edu.cn).

J. C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cuhk.edu.hk).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org. This consists of a 63.3-KB PDF containing Appendixes A–F.

Digital Object Identifier 10.1109/TNET.2016.2594295

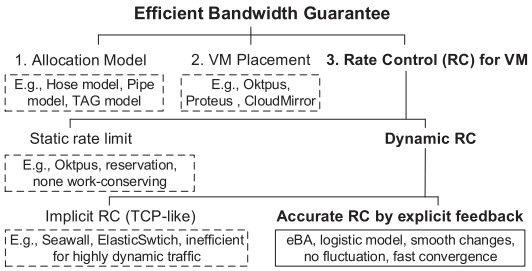


Fig. 1. Efficient bandwidth guarantee requires a bandwidth allocation model, a VM placement algorithm, and a rate control algorithm. *eBA* uses explicit bandwidth information of physical links to enforce accurate rate control for traffic variability in datacenter networks.

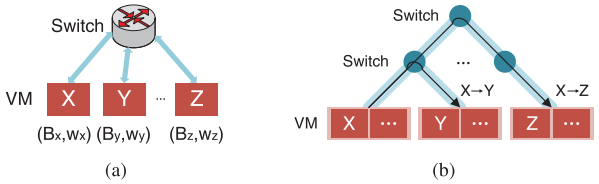


Fig. 2. System model: use hose model for bandwidth requirements of VMs, and allocate bandwidth for VM-pairs on each link based on the requirements. (a) Hose model. (b) Bandwidth allocation model.

guarantee [8]: (i) a bandwidth allocation model to specify the tenants' bandwidth requirements (e.g., hose model [9], Tenant Application Graph [8]), (ii) a VM placement algorithm to place the VMs on servers which can meet the bandwidth requirements in the model (e.g., Oktpus [9]), and (iii) a rate control algorithm to dynamically allocate the bandwidth to achieve both minimum bandwidth guarantee and work-conserving (e.g., Seawall [10]). Previous work on rate control is either static, or based on TCP-like rate control, which deploys multiplicative decrease and makes the throughputs of VMs fluctuating. If there are large numbers of short flows or congestions, the rate control will become unstable and the network utilization may decrease sharply, thus making the bandwidth allocation hard to adapt to the highly variable traffic in datacenters. For example, [9] and [10] use static or time-varying reservation for bandwidth guarantees. They cannot achieve high utilization due to traffic variability in datacenters. References [10], [12], and [13] achieve work-conserving by using TCP-like rate control for VMs. The drawback is that the rate-limit fluctuates when congestions occur, leading to a significant decrease of bandwidth utilization under highly dynamic traffic.

To meet both basic and dynamic requirements of cloud applications, we design *eBA*, an *e*fficient VM *B*andwidth *A*llocation system capable of handling highly dynamic traffic in datacenters. *eBA* provides VM-to-VM bandwidth guarantee by shaping traffic at the VM-level. Unlike TCP-like algorithms, *eBA* uses explicit bandwidth information of physical links to enforce accurate rate control for traffic between VMs, by taking advantage of the low-latency round trip time in datacenters. Such design enables a stable bandwidth allocation, and avoids the performance degradation caused by frequent fluctuations in network throughputs. *eBA*'s rate control function is built upon the Logistic model, which has attractive properties such as fast convergence speed and smooth change rate. In our evaluation, it shows good adaptability to traffic variabilities caused by large number of short flows.

In summary, the contributions of this paper are as follows:

- By considering the characteristics of datacenter network traffic, we present three dynamic requirements of designing an efficient bandwidth allocation algorithm in datacenters.
- We design a distributed rate control algorithm for traffic between VMs based on the Logistic model, and reveal various important properties for this algorithm under the control-theoretic framework, for both the basic requirements and the dynamic requirements.
- We propose traffic-aware adaptation mechanisms to further improve the efficiency and practicability of the logistic-based rate control, and present the design and implementation of *eBA* at the network layer.
- With experiments and trace-driven simulations, we show that the algorithm not only achieves bandwidth guarantees and work-conserving bandwidth allocations, but also adapts to the highly variable traffic in datacenters.

The rest of this paper is organized as follows. Sec. II presents the datacenter network model and the design of rate control functions. In Sec. III, we analyze the model in terms of stability, rate of convergence, and steady state characteristics. A bandwidth allocation algorithm is proposed in Sec. IV. Complementary to our preliminary work [14], we present the design of *eBA* and develop mechanisms for the algorithm to adapt to different traffic patterns in Sec. V and Sec. VI, respectively. Sec. VII evaluates *eBA*. We discuss related work in Sec. VIII, and conclude in Sec. IX.

II. Model and Design

A. Datacenter Network Model

In this paper, we focus on how to allocate the bandwidth to achieve work-conserving bandwidth guarantee under traffic variability in datacenter. Note that work-conserving in network means that if a link is the bottleneck, this link should be fully utilized. We use existing hose model [5] to describe the bandwidth guarantee requirements of VMs, and assume that VM placement is accomplished through an existing approach.

Datacenter Network: We begin by introducing the datacenter network model. We consider a virtualized datacenter, where applications are running in VMs. Let $\mathbb{K} = \{1, 2, \ldots, K\}$ be the set of K VMs in the datacenter. Since we aim at guaranteeing bandwidth for applications at a VM-level, all the traffic between a pair of VMs is viewed as a *VM-flow*. The active VM-flows in datacenter networks constitute a subset of directed VM-pairs, thus $\mathbb{N} \subseteq \{i_{x\to y} \mid x, y \in \mathbb{K}\}$ where VM-flow i is from VM x to y. We number them in order, $\mathbb{N} = \{1, 2, \ldots, N\}$, and denote the rate of VM-pair i at time t as $v_i(t)$, $i \in \mathbb{N}$. The nodes (servers and switches) are connected by physical links and we use $\mathbb{M} = \{1, 2, \ldots, M\}$ to denote the set of M links across the datacenter network. Let C_l be the bandwidth of link l, $l \in \mathbb{M}$.

Bandwidth Requirements of VMs: As shown in Fig. 2(a), the *hose model*, in which all VMs are connected by a

non-blocking switch with dedicated links, specifies the bandwidth requirements of VMs. We use throughput B and weight w for each VM to present the requirements of minimum bandwidth guarantee and proportional bandwidth sharing, respectively. Both B and w are given when a VM is created in the datacenter. Consequently, the cloud operator should guarantee the minimum bandwidth of VMs when there is unsatisfied traffic demand, and achieve a proportional sharing based on their weights when the demands exceed B.

Bandwidth Allocation Model: However, datacenter topology, which in most current datacenters is like a multi-root tree (e.g., [15]), is far more complicated than the hose model. The bandwidth allocation model, which represents the bandwidth allocation on physical links, should consider the fact that the VM-to-VM traffic goes through a number of physical links, and the rate of traffic depends on the congested one. The uncertainty of the congested link forces us to allocate the bandwidth according to each link along a VM-to-VM path, as shown in Fig. 2(b).

We first transform the bandwidth requirements of VMs to VM-flow performance metrics. Let P_l denote the set of VM-flows across link l, $\forall l \in \mathbb{M}$, and L_i denote the set of links passed by VM-flow i, $\forall i \in \mathbb{N}$. The guarantees and weights of VMs can be partitioned into VM-flows' guarantees and weights. For VM-flow i (from VM x to VM y), suppose $\mathbb{V}_x^{out}$ is the set of VMs receiving data from x, and $\mathbb{V}_y^{in}$ is the set of VMs sending data to y. The weight of flow i is expressed as

$$w_i = \frac{w_x}{|\mathbb{V}_x^{out}|} + \frac{w_y}{|\mathbb{V}_y^{in}|}, \tag{1}$$

where $|\mathbb{V}_x^{out}|(|\mathbb{V}_x^{in}|)$ is the number of VMs in $\mathbb{V}_x^{out}(\mathbb{V}_x^{in})$. This way, the shared bandwidth of a cloud tenant, which is the total bandwidth of its rented VMs, will be proportional to the total weights of its VMs (for details, we refer the readers to our previous work [16]).

The bandwidth guarantee of VM-flow i can be obtained by partitioning x's egress bandwidth or y's ingress bandwidth based on their weights [17], thus

$$B_i = \min\{B_x \frac{w_y}{\sum_{z\in\mathbb{V}_x^{out}} w_z}, B_y \frac{w_x}{\sum_{z\in\mathbb{V}_y^{in}} w_z}\}, \tag{2}$$

where $\sum_{z\in\mathbb{V}_x^{out}} w_z$ is the total weight of VMs receiving data from x, and $\sum_{z\in\mathbb{V}_y^{in}} w_z$ is the total weight of VMs sending data to y. The rate of a VM-flow is limited by the congested VM (either the source VM or the destination VM), hence we choose the smaller one between the source and destination bandwidth in case that the total bandwidth guarantee exceeds the link capacity and causes guarantee failure.

B. Logistic Model for Objectives

Considering the dynamic nature of datacenter networks and the shortcomings of previous solutions, our intuition is to use a *smooth* but *fast-growing* function as the rate, with gradual change rate at the beginning and around the equilibrium. This is exactly the situation in ecology where the dynamics of population $x(t)$ is modeled by the *Logistic model* [18] as:

$$\dot{x}(t) = rx(t)(1 - \frac{x(t)}{C}). \tag{3}$$

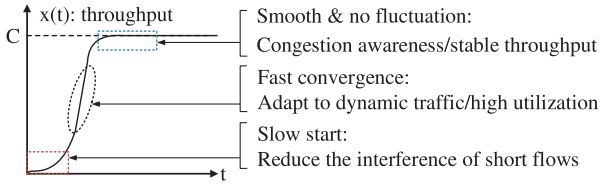


Fig. 3. Key mechanisms and benefits of the proposed datacenter rate control via the Logistic model.

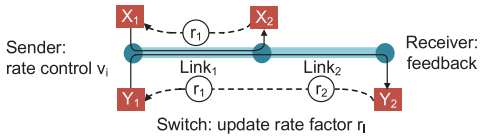


Fig. 4. *eBA* uses explicit bandwidth information of links to control the rate of VM-flows at the source VM: (i) the switch maintains the status of link bandwidth using a rate factor (r_l), (ii) the receiver feeds back the rate factor along a VM-flow's path to the sender, and (iii) the sender controls the rate of VM-flows (v_i) based on this rate factor.

r is the intrinsic rate of increase and C is the resource capacity denoted by the number of organisms. Note that the growth of population is proportion to the population itself, and has an inverse correlation to the resources.

As shown in Fig. 3, a Logistic-like rate controller can benefit the datacenter network from the following aspects: (i) The rates of VM-flows can be controlled according to the available bandwidth, and maintain at the equilibrium (maximum value) with no fluctuation. It can avoid severe decreases when congestions occur. (ii) The rate has an exponential convergence speed, which can quickly adapt to the highly variable traffic demands in datacenters. (iii) With a slow start speed, the bursty traffic caused by massive short flows can be flattened, and traffic fluctuation on bottleneck links can be reduced.

C. Design of Rate Allocation Functions

However, the Logistical model cannot be directly applied to VM rate control, since it has a static upper bound (i.e., C) while we need dynamic rate limits for VM-flows. To address this challenge, *eBA* uses the explicit bandwidth information of physical links to enable bandwidth-aware rate control at source VM. As shown in Fig. 4, the design should consist of two aspects: (i) a *link bandwidth function* on the switch to calculate the link bandwidth that can be used by each VM-flow, and (ii) a *rate control function* on the source VM to update the rates of VM-flows based on the link bandwidth. The receiver can feed back the bandwidth information along a VM-flow's path to the source VM.

Link Bandwidth Function: We first define a *rate factor* for each link, whose differential is:

$$\frac{\mathrm{d}r_l(t)}{\mathrm{d}t} = \beta r_l(t)\left(1 - \frac{\sum_{i\in P_l} v_i(t)}{C_l}\right), \tag{4}$$

where $\sum_{i\in P_l} v_i(t)$ represents the total throughput on link l, $r_l(t) \in [0, C_l]$.

Note that the rate factor increases if there is any available bandwidth, decreases when the allocated bandwidth exceeds

TABLE I
COMMONLY USED NOTATIONS IN THIS PAPER

Notation	Meaning
$v_i(t)$	rate of VM-flow i
L_i	set of links on VM-flow i
$r_l(t)$	rate factor of link l
C_l	capacity of link l
P_l	set of VM-flows across link l
$\widetilde{r}_i(t)$	minimum rate factor in L_i

the link capacity C_l, and becomes stable when the link is fully utilized. Hence, we can use the rate factor $r_l(t)$ to control the rate of VM-flow i at the source VM. This way, the rate factor $r_l(t)$ of a fully utilized link can be regarded as the maximum shared bandwidth that can be used by each VM-flow passing through this link. We choose the minimum rate factor along a VM-flow's path, since the rate of the VM-flow is limited by the most congested link, i.e., $\widetilde{r}_i(t) = w_i \min_{l\in L_i}\{r_l(t)\}$.

Rate Control Function: We can express the differential of VM-flow's rate $v_i(t)$ as:

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \alpha v_i(t)\left(\ln \widetilde{r}_i(t) - \ln v_i(t)\right). \tag{5}$$

We multiply the rate factor with the corresponding weight w_i, which ensures the objective rate of each VM-flow is in proportional with the weight, so as to achieve proportional bandwidth share among VMs. The objective rate of a VM-flow is the bandwidth resource we should allocate to this VM-flow when the rates of all VM-flows get stable.

Note that Eq. (5) is derived by taking the logarithm of the original $1-\frac{x}{C}$ in the Logistic model. The reason is that we can quickly adapt the rate of VM-flow to the received rate factor within $O(\log\log n)$ time (as we will prove in Sec. III-C), thereby improving the convergence speed.

Notations are summarized in Table I. The *rate control model* in datacenter networks can be summarized as:

$$\begin{cases} \dfrac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \alpha v_i(t)\left(\ln \widetilde{r}_i(t) - \ln v_i(t)\right), & \forall i \in \mathbb{N} \\ \widetilde{r}_i(t) = w_i \min\{r_l(t) \mid l \in L_i\} \\ \dfrac{\mathrm{d}r_l(t)}{\mathrm{d}t} = \beta r_l(t)\left(1 - \dfrac{\sum_{i\in P_l} v_i(t)}{C_l}\right), & \forall l \in \mathbb{M}. \end{cases} \tag{6}$$

Remark: Note that $v_i(t)$ and $r_l(t)$ are coupled and form a feedback system. The rate factor is limited by the physical bandwidth and controlled by the throughput of VM-flows. The rate of each VM-flow is limited and controlled by the rate factor. Depending on the interaction between $v_i(t)$ and $r_l(t)$, the system will converge to an equilibrium where the bandwidth is fully utilized. The convergence process corresponds to the population dynamic of the Logistic model.

III. ANALYSIS VIA CONTROL-THEORETIC FRAMEWORK

In this section, we analyze the rate control model via the control-theoretic framework. The analysis assumes that the topology and connections between VMs remain unchanged. We will provide an algorithm to handle the changes of connections in Sec. V-B.

A. Stability of Equilibrium

Since the differential equations in Eq.(6) form a nonlinear system, we use the Lyapunov stability theory to prove its stability. For the nonzero equilibrium of the system, let $\mathbb{M}' \subseteq \mathbb{M}$ be the set of bottleneck links. We have:

Theorem 1: For $v_i(t), r_l(t) \in (0,\infty]$, where $i \in \mathbb{N}, l \in \mathbb{M}'$, the system formulated by Eq. (6) is *locally asymptotically stable* irrespective of capacities of bottleneck links and communication pattern of VMs.

Proof: We begin the proof with substitutions. Let $\varphi_i(t) = \ln v_i(t)$ and $\phi_l(t) = \ln r_l(t)$, then Eq. (6) can be derived as

$$\begin{cases} \dfrac{\mathrm{d}\varphi_i(t)}{\mathrm{d}t} = \alpha(\widetilde{\phi}_i(t) - \varphi_i(t)), \\ \widetilde{\phi}_i(t) = \ln w_i + \min\{\phi_l(t) \mid l \in L_i\}, \\ \dfrac{\mathrm{d}\phi_l(t)}{\mathrm{d}t} = \beta\left(1 - \dfrac{\sum_{i\in P_l}\exp\{\varphi_i(t)\}}{C_l}\right). \end{cases} \tag{7}$$

Let $f_l(\varphi_i(t))$ denote the function of $\varphi_i(t), i \in P_l$, i.e., $\frac{\mathrm{d}\phi_l(t)}{\mathrm{d}t} = \beta f_l(\varphi_i(t))$. Note that $f_l(\varphi_i(t))$ is continuous and integrable. We define a positive function $F_l = \int_0^{-f_l(\varphi_i(t))} x\mathrm{d}x$ in the neighborhood of the equilibrium. Then we have $\frac{\partial F_l}{\partial \varphi_i} = -f_l(\varphi_i(t))$. To apply the Lyapunov stability theory, we first construct a positive function for all VM-flows ($i \in \mathbb{N}$) and all bottleneck links ($l \in \mathbb{M}'$):

$$V(\varphi_i(t),\phi_l(t)) = \alpha\sum_{i\in\mathbb{N}}\frac{1}{2}\left(\widetilde{\phi}_i(t) - \varphi_i(t)\right)^2 + \beta\sum_{l\in\mathbb{M}'}F_l(\varphi_i(t)).$$

For those links that are not bottlenecked, since their bandwidth are not fully utilized and the derivative of r_l is above 0, the rate factor will continuously increase even when the rates of all VM-flows get stable. However, as the rate of VM-flows are limited by the bottleneck links, these under-utilized links or non-convergent rate factors will not impact the rate of VM-flows. Hence, the bandwidth allocation can converge irrespective of non-bottleneck links.

We have the partial differential of V on $\varphi_i(t)$ and $\phi_l(t)$.

$$\frac{\partial V}{\partial \varphi_i} = -\alpha(\widetilde{\phi}_i(t) - \varphi_i(t)) - \beta\sum_{l\in L_i^*} f_l(\varphi_i(t)),$$

$$\frac{\partial V}{\partial \phi_l} = \alpha\sum_{i\in P_l^*}(\widetilde{\phi}_i(t) - \varphi_i(t)),$$

where $P_l^* \subseteq P_l$ is the set of VM-flows bottlenecked on link l ($\widetilde{\phi}_i(t) = \ln w_i + \widetilde{\phi}_l(t)$), and $L_i^* \subseteq L_i$ is the set of bottleneck links passed by VM-flow i ($l \in L_i \cap \mathbb{M}'$). Then we can derive the differential of V with respect to time variable t as (refer to Appendix A of supplementary materials for derivation):

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\sum_{i\in\mathbb{N}}\alpha^2(\widetilde{\phi}_i(t) - \varphi_i(t))^2 \le 0.$$

Note that $\dot{\varphi}_i(t) = \dot{\phi}_i(t) = 0$ is the equilibrium of the system in Eq. (7). The function V is positive and the differential of V is negative except for the equilibrium. Based on Lyapunov's second method for stability [19], function V is a Lyapunov-candidate-function of the system and the equilibrium is proven to be locally asymptotically stable. ■

Insight: For VM-flows with given initial rates (> 0), they will converge to an equilibrium under the control of our model. The convergence depends on neither the link bandwidth in datacenters, nor the placement and communication patterns of VMs. The rate control for each VM-flow is completely independent, and can easily be scaled up via a distributed implementation. The following discussion is based on the fact that the system is asymptotically stable.

B. Achieving Efficiency and Fairness

We first present the definition of *max-min fairness* in the context of VM bandwidth sharing. The bandwidth allocation is called max-min fairness if the allocation is feasible, and for each VM-flow we cannot increase its rate without decreasing the rates of other VM-flows whose rates are equal to or less than this flow across its bottleneck link [20]. Note that max-min fairness indicates that the bandwidth allocation is also work-conserving.

Theorem 2: In a multiple-bottleneck topology where N VM-flows share M links, if there exists a unique equilibrium, then the rates of VM-flows allocated by the algorithm in Eq. (6) achieve the *weighted max-min fairness*. (Refer to Appendix B of supplementary materials for the proof.)

Insight: Our model can fully allocate the bandwidth of a bottleneck link to the VM-flows across it. This allocation result is a proportional sharing among VM-flows according to their weights. It indicates that the model meets the static requirements of work-conserving and network proportionality [5] in datacenter networks. For bandwidth guarantee, we will introduce a threshold based method in Sec. IV.

C. Rate of Convergence

Convergence to Efficiency: We first propose a function to quantify the network utilization.

Definition 1: Given a constant $\lambda \in (0, 1]$, when n VM-flows are sharing a single bottleneck link l, the resource allocation is called λ efficiency if there exists time T_λ such that for $t > T_\lambda$

$$h(t) = \frac{\sum_{i=1}^{n} v_i(t)}{C_l} \geq \lambda. \tag{8}$$

Let $t = 0$ be the start time where $v_i(0) = v_0 \ll C_l$ (v_0 is much less than C_l). The minimum T_λ is called the time of convergence to efficiency, thus $h(T_\lambda) = \lambda$.

The efficiency definition describes the utilization of the congested link when the system converges to the equilibrium. To qualify the time of convergence to efficiency, we consider a scenario where n homogeneous VM-flows are sharing the same bottleneck link with C_l capacity.

Proposition 1: Given n VM-flows across the same bottleneck link, the time of convergence to λ efficiency satisfies:

$$T_\lambda < \frac{1}{\alpha} \ln \frac{\ln (nv_0/C_l)}{\ln \lambda} + \frac{1}{\beta} \ln \left((\frac{C_l}{nv_0} - 1) \frac{\lambda}{1 - \lambda} \right).$$

Refer to Appendix C of supplementary materials for the proof.

Convergence to Fairness: We start with defining a fairness function.

Definition 2: Given a constant $\rho \in (0, 1]$, when n VM-flows are sharing a single bottleneck l, the resource allocation is called ρ fairness if there exists time T_ρ such that for $t > T_\rho$

$$g(t) = \frac{x_j(t)}{x_k(t)} \geq \rho, \tag{9}$$

where $x_j(t) = v_j(t)/w_j, j \in \arg\min_{i \in P_l} v_i(t)$ and $x_k(t) = v_k(t)/w_k(t), k \in \arg\max_{i \in P_l} v_i(t)$. $g(t)$ is the ratio of the minimum normalized rate to the maximum normalized rate. Let $t = 0$ be the start time where $v_i(0) = C_l/n$, and the minimum value T_ρ is called the time of convergence to fairness, thus $g(T_\rho) = \rho$.

The fairness definition describes the gap between the maximum and minimum normalized rates. To obtain the time of convergence to fairness, we assume that the rates of n homogeneous VM-flows have reached the equilibrium at the shared bottleneck with C_l capacity before another VM-flow v_j joins in.

Proposition 2: Given n VM-flows across the same bottleneck link at a stable state, the time of convergence to ρ fairness with a new VM-flow satisfies the following equation

$$T_\rho = \frac{1}{\alpha} \ln \frac{\ln (nv_0/C_l)}{\ln \rho}.$$

Refer to Appendix D of supplementary materials for the proof.

Insight: Definition 1 and 2 correspond to two common scenarios in datacenters: the utilization of a link suddenly peaks with the arrival of many flows, and a highly utilized link becomes congested as a new flow joins in. The burstiness of traffic and the short inter-arrival time of flows (average at 15ms [6]) require swift response of the rate control. Our model has $O(\log \log n)$ convergence time on the accuracy factors λ and ρ, for both convergences to efficiency and fairness. Hence, it is suitable for datacenter traffic.

D. Steady State Characteristics

The performance of the system in the neighborhood of steady state is characterized in this section. In the control theory, a nonlinear system can be considered linear about the equilibrium for small changes Δv and Δr [21]. By using the state space approach ([21, Sec. 2.3]), we can obtain a linear approximation of the system in Eq. (6), which can be further analyzed by Laplace transform.

Let $\mathbf{v}(t)$ denote the rate vector of VM-flows, thus $\mathbf{v}(t) = [v_1(t), v_2(t), \ldots, v_n(t)]$, and $r(t)$ be the rate factor of the bottleneck link. Based on Eq. (6), we define $F_i(\mathbf{v}, r)$ $(i = 1, 2, \ldots, n)$ and $H(\mathbf{v}, r)$ as the functions of $\mathbf{v}(t)$ and $r(t)$:

$$\begin{aligned} F_i(\mathbf{v}, r) &= \alpha v_i(t)(\ln w_i r(t) - \ln v_i(t)), \\ H(\mathbf{v}, r) &= \beta r(t)(1 - \frac{\sum_{i \in P_l} v_i(t)}{C_l}). \end{aligned} \tag{10}$$

Suppose $O = (\mathbf{v}^*, r^*)$ is the operating point, where $v_i^* = w_i r^*$ and $r^* = C/\sum_{i=1}^{n} w_i$. Let $\Delta v_i = v_i - v_i^*$ and $\Delta r = r - r^*$ be the small changes of state. The linear

differential equations for Δv_i and Δr can be derived as (refer to Appendix E of supplementary materials for detailed transformations)

$$\begin{cases} \Delta\dot{v}_i = -\alpha\Delta v_i + \alpha w_i \Delta r, \ i \in \mathbb{N}, \\ \Delta\dot{r} = \beta \sum_{i=1}^{n} \dfrac{\Delta v_i}{\sum_{j=1}^{n} w_j}. \end{cases} \tag{11}$$

Note that in the neighbourhood of O, each rate Δv_i can be denoted as $\Delta v_i = w_i \Delta v$, where Δv belongs to the same domain with Δv_i. By replacing v_i with $w_i \Delta v$ in Eq. (11), we find that all the equations for $\Delta\dot{v}_i$ can be derived as a group of linearly dependent equations. Hence, the linear system can be simplified as

$$\begin{cases} \Delta\dot{v} = -\alpha\Delta v + \alpha\Delta r, \\ \Delta\dot{r} = \beta\Delta v, \end{cases} \tag{12}$$

which is a second-order system. The damping ratio of the system can be derived as $\zeta = \frac{1}{2}\sqrt{\alpha/\beta}$.

To solve the relationship between α and β, we first consider the stability of the system. Note that when $s = j\omega$, the phase satisfies

$$\angle G(j\omega) = -\pi + \arctan\frac{\alpha}{\omega} \subseteq (-\pi, -\frac{\pi}{2}), \tag{13}$$

thus the system is natively stable. We then consider the response of the system ([21, Sec. 5.3]): (i) the swiftness of response, and (ii) the closeness of the response to the desired response. Generally, ζ is set as a value from 0.5 to 1. As ζ decreases, the system becomes more swift while the overshoot of rate becomes larger. Since they are contradictory requirements, a compromise must be obtained.

Insight: The characteristics of steady state imply the situation where steady long flows suffer interferences from large numbers of short flows, congestions, inaccurate rate enforcement, etc. When small changes occur in the system, we want the rate to swiftly respond to the changes and converge to another equilibrium. However, this will increase the overshoot of rate, and cause strenuous variation to the rate of VM-flow. To smooth the overshoot as well as to maintain the ability of quick recovery, we maintains ζ at around 0.5.

IV. Algorithms Design in Datacenters

A. Discretization on Time Series

As shown by the design goal in Sec. II, the rate control for each VM-pair is distributed and suitable for execution in VMs. The main issue is how to discretize the continuous functions into equivalent approximate recursions that can be implemented in datacenter networks.

We use τ_v and τ_r as the time interval to update $v_i(t)$ and $r_l(t)$, respectively. This way, the time variables can be denoted as $k\tau_v$ and $k\tau_r$, where $k \in \{0, 1, 2, \ldots\}$ is the update rounds. For brevity, we use $v_i(k)$ and $r_l(k)$ to represent $v_i(k\tau_v)$ and $r_l(k\tau_r)$. The continuous functions are then transformed into time-discrete functions. The approximation can be derived as:

$$\frac{\mathrm{d}\ln v_i(t)}{\mathrm{d}t} \approx \frac{\ln v_i(k+1) - \ln v_i(k)}{\tau_v}, \tag{14}$$

and the same is true for $r_l(t)$.

Substituting Eq. (14) into Eq. (6) and replacing the time continuous functions with $v_i(k)$ and $r_l(k)$ yield recursion $\{v_i(k)\}$ and $\{r_l(k)\}$ as

$$\begin{cases} v_i(k+1) = v_i(k)^{1-\alpha\tau_v}\big(r_i(\widetilde{k_r})\big)^{\alpha\tau_v}, \\ r_l(k+1) = r_l(k)\exp\left(\beta\tau_r(1 - \dfrac{\sum v_i(\widetilde{k_v})}{C_l})\right), \\ r_i(\widetilde{k_r}) = w_i \min\{r_l(\widetilde{k_r}) \mid l \in L_i\}. \end{cases} \tag{15}$$

Since the updates of $v_i(k)$ and $r_l(k)$ are asynchronous, each with their respective cycles, we use $\widetilde{k_r}$ to represent the corresponding rate factor when controlling the rates of VM-flows. Similarly, $\widetilde{k_v}$ is the exact rate that can be measured by a switch. Considering the delay τ_d of receiving rate factors for a VM-flow, we have $\widetilde{k_r} = \lfloor\frac{k\tau_v - \tau_d}{\tau_r}\rfloor$ and $\widetilde{k_v} = \lfloor\frac{k\tau_r}{\tau_v}\rfloor$.

In fact, in the implementation, $v_i(\widetilde{k_v})$ and $r_l(\widetilde{k_r})$ can be obtained without the knowledge of $\widetilde{k_v}$ and $\widetilde{k_r}$. The VM's rate controller can use the latest received rate factor, and the switch only needs to obtain the throughput on each link.

B. Minimum Guarantee

The above discrete rate control functions provide efficient bandwidth allocation in weighted max-min fairness. To ensure minimum bandwidth guarantee for each VM-flow, we set the bandwidth guarantee as a lower bound for the rate limit of each VM-flow. When the rates of VM-flows in the recursions (Eq. (15)) are lower than the bandwidth guarantees, their rate limits will still be the guaranteed bandwidth. Hence, we use another variable $v_i^{\max}(k)$ as the rate limit for VM-flow, which is represented as

$$v_i^{\max}(k) = \max\{B_i, v_i(k)\}. \tag{16}$$

This way, if the VM-flow's bandwidth demand is larger than the bandwidth guarantee, it will be ensured with the guaranteed bandwidth. When the VM-flow's traffic demand is lower than the bandwidth guarantee, the underutilized guarantee leads to remaining bandwidth on the congested link. According to the function in Eq. (15), the rate factor will increase consequently. When other VM-flows on this link receive the increased rate factor, their rate limits will continue to increase until fully utilizing the remaining bandwidth if they have any unsatisfied traffic demands. In Appendix F of the supplementary file, we use an example to show how to achieve minimum guarantee.

One question is how to control the infinite increase of the rate factor, which happens on any other uncongested links. The solution is to set an upper bound for each rate factor on the switches. As the received rate factor (r_l) is the upper bound of the rate limit (v_i) of corresponding VM-flow, the rate limit becomes unavailable when exceeding the physical link capacity. Hence, we maintain r_l below the physical link capacity by using an upper bound:

$$r_l(k) = \min\{C_l, r_l(k)\}. \tag{17}$$

With the above constraints in Eq. (16) and (17), we can provide bandwidth guarantees for VMs as well as maintain work-conserving.

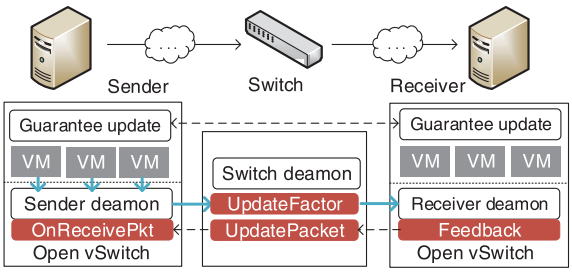


Fig. 5. The design of *eBA* consists of two functions: guarantee update and rate control. The implementation of rate control relies on three modules: sender, switch and receiver modules, which form a closed loop for each VM-flow.

Algorithm 1 Bandwidth Guarantees Update on VM x

Input: Set of VMs with destination x: $\mathbb{V}_x^{in} = \varnothing$
Set of VMs with source x: $\mathbb{V}_x^{out} = \varnothing$
Sum of weights for VMs in $\mathbb{V}_x^{in}$: $S_x^{in} = 0$
Sum of weights for VMs in $\mathbb{V}_x^{out}$: $S_x^{out} = 0$
VM's bandwidth guarantee and weight: (B_x, w_x)
Output: Bandwidth guarantees for VM-flows on x: $B_{x\to y}$

function **UpdateGuarantee**(weight w_y, w_x)
  for $i \in \mathbb{V}_x^{out}, x \in \mathbb{V}_y^{in}$ **do**
    $B_{x\to y} = \min\{B_x^{out}\frac{w_y}{S_x^{out}}, B_i n^{in}\frac{w_x}{S_x^{int}}\}$
    $w_{x\to y} = w_x/|\mathbb{V}_x^{out}| + w_y/\mathbb{V}_y^{in}$
  end for
end function
function **OnEstablishConnection**(VM y)
  $\mathbb{V}_x^{out} = \mathbb{V}_x^{out} \cup y$
  $\mathbb{V}_y^{in} = \mathbb{V}_y^{in} \cup x$
  $S_x^{out} = S_x^{out} + w_y$
  $S_y^{in} = S_y^{in} + w_x$
  UpdateGuarantee(w_y, w_x)
end function
function **OnCloseConnection**(VM y)
  $\mathbb{V}_x^{out} = \mathbb{V}_x^{out} - y$
  $\mathbb{V}_y^{in} = \mathbb{V}_y^{in} - x$
  $S_x^{out} = S_x^{out} - w_y$
  $S_y^{in} = S_y^{in} - w_x$
  UpdateGuarantee(w_y, w_x)
end function

Algorithm 2 Rate Control for VM-Flow $x \to y$

Input: Parameter settings: α, β, τ_v, τ_r
Bandwidth guarantee for VM-flow $x \to y$: $B_{x\to y}$
Output: Rate limit on VM-flow: $v_{x\to y}$

function **SendPkt**(Packet pkt, τ_v)
  Sleep for τ_v
  $pkt.identifer$ = 0
  $pkt.r = C_l$
end function *//sender*
function **OnReceivePkt**(Packet pkt, receiver y)
  $v_{x\to y} = v_{x\to y}^{1-\alpha\tau_v} * (w_{x\to y} * pkt.r)^{\alpha\tau_v}$
  $v_{\max} = \max\{B_{x\to y}, v_i(k)\}$
  Set rate limit $v_{\max}$ to VM-flow $x \to y$
end function *//sender*
function **Feedback**(Packet pkt, sender x)
  $pkt.identifer$ = 1
  Feedback Packet pkt to sender x
end function *//receiver*
function **UpdateLinkFactor**(time τ_r, link l)
  Sleep for τ_r
  Get aggregate rate V_l for link l
  $r_l = r_l * \exp\left\{\beta\tau_r(1 - \frac{V_l}{C_l})\right\}$
  $r_l = \min\{r_l, C_l\}$
end function *//switch*
function **UpdatePacket**(Packet pkt, link l)
  if $r_l < pkt.r$ and $pkt.identifer$ = 0 **then**
    $pkt.r = r_l$
  end if
Send packet out
end function *//switch*

V. *eBA* Design

A. Design Overview

We now describe the design of *eBA* according to the abstraction in Fig. 5. Recall that *eBA* enforces VM-pair rate control with given VM performance metrics. Hence, *eBA* design consists of two main functions: First, the guarantee update which transforms VM performance metrics to VM-flow performance metrics based on Algorithm 1. Second, the rate enforcer which limits the sending rate of each VM-flow based on Algorithm 2.

The rate control, as indicated by our model in Eq. (15), is a control loop involving all the elements along the path of a VM-flow. Hence, the implementation of rate control needs the cooperation of modules on sending servers, receiving servers and switches. Specifically, the sending servers are responsible for enforcing the rate limit $v_i(k+1)$ in Eq. (15), and the switches maintain the rate factor $r_l(k+1)$ in Eq. (15).

B. Bandwidth Guarantee Update

To avoid the limited scalability in centralized computation for mass of VM-pairs, *eBA* leverages a distributed manner to update guarantee under the cooperation of servers. Each server is equipped with a guarantee update module, which requires weight/guarantee of VMs on other servers and echoes the request for weight/guarantee of VMs on this server. The local deployment also has the advantage on monitoring the changes in VM connections with low latencies.

Apparently, bandwidth guarantees in datacenters for VM-flows can be variable. When a VM establishes a connection to another VM, or closes the connection to a VM, the bandwidth guarantees of VM-flows should be updated. *eBA* captures such changes and updates bandwidth guarantees as the process shown in Algorithm 1. The update is triggered by two events: (i) connecting to a new VM, (ii) disconnecting from a VM. Specifically, when VM x establishes connections to VM y which has no connection with x previously, the *OnEstablishConnection* function will add y to the set of VMs receiving from x, and add x to the set of VMs sending to y. In the end, the algorithm updates the bandwidth guarantees and weights for all VM-flows on VM x and y according to

Eq. (1) and (2). Similarly, when VM x closes the connection to a VM, the *OnCloseConnection* function will update the bandwidth guarantees and weights for all VM-flows on VM x and y.

eBA calculates the guarantees for the minimum set of VM-flows associated with the changed connection to reduce the overhead in guarantee update. Given that average number of servers communicating with each server are small scale (about 6 in data centers with 1500 servers [6]), the computation can finish within very short period, which is considered efficient enough to input for the rate control algorithm.

C. Rate Control Framework

Based on Sec. IV, the key in the Logistic based Rate Controller (**LRC**) is to collect the minimum rate factor along a VM-flow's path. To reduce the traffic overhead of control message, *eBA* sends a *control packet* to traverse the links passed by a VM-flow. The control packet is assigned with the same destination as the corresponding VM-flow. Since the control packet and data packet in the VM-flow share the same matching rules on every passed switch, the control packet will go through the same set of links as the VM-flow. The control packet is then updated by the switch deamon and fed back by the receiver deamon. As shown in Fig. 5, *eBA* consists of three modules: (i) sender deamon which constructs the control packet, (ii) the switch deamon which updates the rate factor on the packet, and (iii) the receiver deamon which feeds back the packet.

Rate Control Loop: According to Eq. (16), the rate control for each VM-flow relies on the output (bandwidth guarantee) of Algorithm 1 and the distributed recursions in Eq. (15). The update for rate of VM-flow $x \rightarrow y$ can be accomplished in a cooperative manner as shown in Algorithm 2. For one VM-flow, the sender (for VM x) periodically sends a *control packet* to the receiver (for VM y), every time interval τ_v. The switch updates the link factors with the measured aggregate rate based on Eq. (15) every τ_r. After receiving a control packet, the switch will update the rate factor if the packet passes through a link with a lower rate factor. The receiver then feeds back the packet to the sender without changing the rate factor. This way, when the packet arrives at the sender, the rate factor will be the smallest one along its path. For the sender, the *OnReceivePkt* function will be called each time it receives a control packet. The function updates the rate of VM-flow according to Eq. (15), and enforces the rate limit to VM-flow $x \rightarrow y$.

Control Packet: Since the rate control works at the VM level, *eBA* uses network layer packet to provide VM-pair (which can be specified by a pair of IP addresses) oriented control, Thus, the packet has the same source and destination IP addresses as a corresponding VM-flow. As the direction of traffic, the rate factor also has two directions on each link. To ensure the rate factor have the same direction as the corresponding VM-flow, we need a binary variable as an *identifier* for the direction of the control packet, and the identifier cannot be modified on the backward path. For the rate factor, we use a 32bit integer to represent it (in Bytes per second). We implement the control packet by filling the data segment of an ICMP packet after the IP header and ICMP header. In ICMP, this data segment will be fed back to sender without any changes, which is suitable for *eBA*'s rate control loop.

Rate Control at Edge: The sender deamon is located on each host server which is capable of enforcing the rate of VM-flows. The construction of control packets is periodically executed every τ_v by $SendPkt$. When the $OnReceivePkt$ function is triggered by receiving a control packet, the sender deamon re-calculates the rate limit by using the received rate factor and the measured time interval between two updates, and then enforces the rate limit to the corresponding VM-flow. The delivery and reception of control packet are asynchronously executed by two separated processes, to avoid that the reception of control packet is suspended due to any packet loss.

Rate Factor at Switch: To reduce the overhead of maintaining the rate factor at the switch, we apply a configurable time interval to each port to periodically obtain the bytes received by this port ($UpdateLinkFactor$ in Algorithm 2). The time interval can be used to control the overhead of continuous update of rate factor. We only monitor the receiving rate of the port since each link connects with two ports, thus the utilization of bandwidth on both directions can be obtained at the receiving end. Correspondingly, the modification of the rate factor in a packet is processed ($UpdatePacket$ in Algorithm 2) when the packet is received and classified as an ICMP packet with the identifier. As the control packet needs to feedback the rate factor of link which has the same direction with the VM-flow, the switch only modifies the forwarding packets and leaves the feedback ones unchanged. Considering hardware in datacenter are highly customized and the operation in switches is simple (a switch only needs to attach the same information to few control messages), the switch deamon can be easily realized.

TABLE II
TUNABLE ALGORITHM PARAMETERS FOR ADAPTATION OF TRAFFIC VARIABILITY

Parameters	Impact on performance
Base bandwidth B	Minimum bandwidth guarantee for VMs
Weight w	The ratio of sharing the residual bandwidth
Update interval τ_r	The precision of rate factor on switch
Update interval τ_v	The precision of rate control on host
Coefficient α, β	The rate of convergence and the overshoot in convergence process

VI. TRAFFIC-AWARE ADAPTATION

The parameters in the rate control model have significant impacts on the performance of the algorithm. To enable a practical implementation of *eBA* for datacenter applications, there is a need to examine how these parameters may impact the algorithm efficiency, and further develop self-adaption mechanisms for *eBA* to efficiently work with different traffic patterns. The parameters are summarized in Table II.

A. Parameters and Tradeoffs

VM Parameters: B and w. VM parameters are provided by tenants as the network performance requirements of VMs. Since the equilibrium of the nonlinear system solely depends

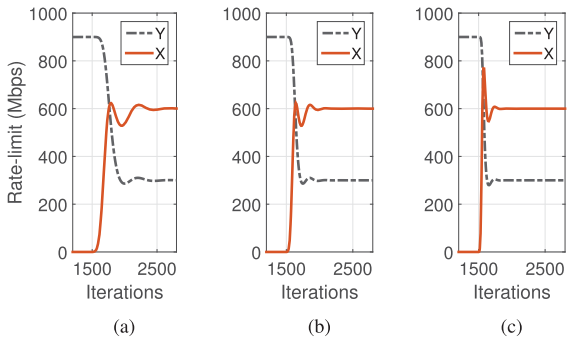


Fig. 6. An example to show the tradeoff between convergence speed and smoothness of the rates of VM-flows: VM-flow X joins in to compete for the bandwidth taken by VM-flow Y under different α, β. We record the rate-limits of X and Y. (a) $\alpha = 1$, $\beta = 2$. (b) $\alpha = 2$, $\beta = 4$. (c) $\alpha = 4$, $\beta = 4$.

on the input bandwidth guarantees and weights of VMs, these two parameters determine the bandwidth allocation result for each VM-pair. By tuning the guaranteed bandwidths and weights, cloud providers are able to flexibly balance the trade-off between *bandwidth guarantee* and *proportional sharing*.

Time Parameters: τ_v and τ_r. The time parameters are used to control the frequency of updating the rate of VM-flows and rate factor of links. One can improve the *precision* of bandwidth allocation by using very small τ_v and τ_r, however such improvement will certainly bring about large *overhead* to the servers and networks. The choice of a suitable period should consider the following facts. First, the interval of updating rate limit should not interfere with the underlying TCP congestion control, hence, τ_v is expected to be an order of magnitude larger than the RTT in datacenters, which is about 1ms [13]. Second, the inter-arrivals of 70% TCP flows from/to servers are periodic at about 15ms [6]. 50% number of flow duration is larger than 10ms, accounting for 99% of the bytes [6]. To keep pace with the variation caused by the short flows, we set τ_v to tens of ms (50ms in our testbed). For flows that last less than 50ms, we can reserve some bandwidth on each link to avoid being congested by other long flows (Sec. VI-B).

For the switches, the frequency of updating rate factors needs to be similar as the rate control at the edge, since the error in the rate-limit of VM-flow will be magnified if we use an inaccurate rate factor. To guarantee the accuracy of rate factor as well as avoid much computational overhead, we set the time period of updating rate factors to $\tau_r \in [\tau_v, 2\tau_v]$.

Convergence Parameters: α and β. These two parameters have impacts on the rate of convergence and the stability of convergence process. According to Proposition 1 and 2, the time of convergence to fairness/efficiency is inversely proportional to α/β. With a constant rate factor, the rate of VM-flow can monotonously increase until it reaches the objective rate. However, in a realistic environment both the rate factors and rates of VM-flows are dynamic due to the changes in bandwidth resources and connections of VMs. The interaction between the dynamics of v and r causes fluctuations to the rate control of VM-flows.

As shown in Fig. 6, we find that there exists a trade-off between the *convergence speed* and the *smoothness* of VM-flow's rate. Though increasing α and β can significantly improve the convergence speed, it will also cause a large overshoot when the rate approaches the equilibrium. The objective of choosing the right values for α and β is to try to maximize the convergence speed while keeping the overshoot under an acceptable ratio, in case of fluctuations. Note that the increment of the rate factor r and the rate of VM-flow v in each iteration are related to β, τ_r and α, τ_v respectively, the setup of these parameters should be jointly considered.

B. Adaptation for Traffic Variability

Capacity Constriction: We apply a ratio $\mu \in (0, 1)$ to the capacity C_l of each link l to keep a positive gap between the link bandwidth and the maximum aggregated rate on the link. As discussed in Fig. 6, the convergence process of r_l is non-monotonic. This indicates that the aggregated rate of all VM-flows on a link may exceed the capacity of this link in Eq. (15), since the decrease of $r_l(k+1)$ is due to $\sum v_i(\widetilde{k_v})/C_l > 1$. However, in the algorithm design, this aggregated rate $\sum v_i(\widetilde{k_v})$ is the throughput measured on the link, which should be less than the bandwidth capacity if there is no error. As a consequence, once the rate factor r reaches at the maximal capacity, it will never converge to the objective value. To enable the convergence of rate factors under real-world context, we need to introduce a threshold capacity (μC_l) within the physical link bandwidth, which is called *Capacity Constriction*. When the threshold capacity is exceeded, the rate factor will decrease so as to notify all the VM-flows passed through to reduce the sending rate. This way, the rate factor is able to converge to a stable point and the maximal link utilization is maintained at μ.

In our testbed, the maximum throughput of TCP flows on 1Gbps links can achieve about 940Mbps. We set μ to 90%, [1] leaving 40Mbps gap between the stable throughput and the link bandwidth. The constricted throughput can benefit the performance of *eBA* from the following aspects:

- $\mu = 0.9$ can guarantee a high bandwidth utilization as well as provide fast convergence for the rate factors.
- The 40Mbps reserved bandwidth reduces the probability of short congestions on links, which can further avoid the losses of control packets.
- *eBA* leverages periodical rate limit to control the throughput of VM-flows, hence, for flows that last less than the update interval (50ms), *eBA* can hardly observe the change in throughput caused by these flows. Such short flows are always latency-sensitive but creating very little traffic in the networks. Hence, reserving 40Mbps bandwidth will avoid the delay and packet loss caused by these bursty flows, and reduce their flow completion times.

Idle Relaxation: The time period τ_r of updating the rate factor is extended to τ_r/ν when link utilization ν is low (e.g., $< 50\%$). As discussed in Eq. (17), when the physical link remains at a low utilization for a number of periods, the rate

[1] μ is an experimental value for our testbed, which guarantees the convergency of LRC as well as a high utilization of physical bandwidth. For 10Gbps Ethernet, the suitable μ also needs to be examined on a real testbed. We leave the examining of the optimal μ on 10Gbps testbed as our future work.

factor will statically stay at C_l. In such situation, since the physical bandwidth is enough to support the traffic demands of VM-flows, there is no need to enforce rate control for VM-to-VM traffic. In datacenters, 97% of the links remain below 70% utilization for more than 99% time period [6]. Starting from this observation, we can reduce the overhead of frequently computing the rate factors on switches by relaxing the time period of updating r, which we call the *Idle Relaxation* strategy. The throughput measurement needs to keep on monitoring any changes in the link utilization. The range of low utilization is set to $[0, 0.5)$ in *eBA*, which aims to give enough iteration steps for r to converge to the objective value under increasing link utilization (> 0.5). Also, when ν is larger than 50%, the reduction in computational overhead becomes minor. Hence, there is no need to extend time period τ_r under this situation.

Uniform Increase: The products $\alpha\tau_v$ and $\beta\tau_r$ remains constant with different time parameters, where $\alpha\tau_v \in (0, 1), \alpha = 4\zeta^2\beta$. Recall the rate of convergence in Sec. III-C, the convergence times are inversely proportional to α and β. With static convergence parameters, one can hardly improve the efficiency of bandwidth allocation even if using more fine-grained rate control, i.e., reducing τ_r and τ_v, which is against the objective of time parameters on improving algorithm efficiency and brings no benefit at the cost of increasing overhead. To address this, we can use dynamic convergence parameters to maintain $\alpha\tau_v$ and $\beta\tau_r$ constantly, since the change of rate limit relies on the product $\alpha\tau_v$ and $\beta\tau_r$ as indicated by Eq. (15). Therefore, when providers prefer accurate bandwidth guarantee, we can increase the frequency of updating the rates of VM-flows at the cost of more computational overhead. On the contrary, if overhead is the key consideration, we can enlarge the time parameters.

C. Implementation Issues and Limitations

Control Packet Overhead: *eBA* maintains control packets for each VM-flow, hence leading to traffic overhead in datacenter networks. Given the 50ms update time interval and the 74Bytes ICMP control packet, each VM-flow can create 1480Bytes/s extra control traffic. Though this is negligible as compared with 1Gbps or 10Gbps bandwidth, *eBA* may still suffer from its linear scalability as the number of VMs increases. Fortunately, the traffic between VM-pairs does not scale with the increase of total VMs. As indicated by [6], 89% server pairs within the same rack never exchange traffic, and the ratio for server pairs across racks is 99.5%. Reference [6] also shows that a server talks to two servers within its rack and four servers outside the rack on average. Hence, the traffic overhead can be maintained within an acceptable ratio ($<$ 1Mbps on average). In addition, the VMs that belong to the same tenant are always placed under the same rack switch or same aggregate switch. Hence, to avoid heavy traffic overhead on the root switches, we can deploy *eBA* under each aggregate switch, without guaranteeing the traffic across the root switch in multi-tenant datacenters.

Losses of Control Packets: *eBA*'s rate control relies on the update of rate limit according to the rate factors from the received ICMP packets. When the losses of control packets occur, the update of rate limit will be held until the arrival of next control packet. However, as indicated by Theorem 1, the system is asymptotically stable and can converge after enough iteration steps. For example, losing one ICMP packet means that the time period between two iterations is doubled, which will not have any impact on the convergence of the system, but prolong the convergence time of the rate allocation to the VM-flows. Fig. 7(a) shows the convergence of *eBA*'s rate control with a severe loss rate of control packets (50%). The throughput of VMs can still converge. In fact, the ping messages between certain server pairs only experience around 2% loss rate in datacenters [6]. With such loss rate, as shown in Fig. 7(b), the rate control of *eBA* is stable enough to offer bandwidth guarantee for VMs.

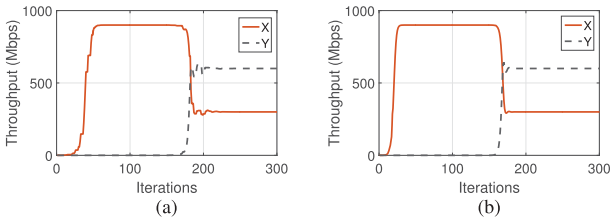


Fig. 7. *eBA*'s rate control with losses of control packets. VM-flow Y joins in to compete for the bandwidth taken by VM-flow X on a physical link. The weights of VMs are $w_X : w_Y = 1 : 2$. (a) 50% loss of control packets. (b) 2% loss of control packets.

Equal-Cost Multi-Path Routing (ECMP): State-of-the-art forwarding in datacenter networks leverages ECMP [22] to balance the inter-server traffic load by hashing flows across multiple available paths. One VM-flow will be split into several links with different available bandwidth. The control packets can hardly be ensured to pass through the same path of the corresponding VM-flow. At present, in order to accurately obtain the available bandwidth for VM-flows, we use per-VM-flow hashing in ECMP by checking the tuple (source IP, destination IP) but ignoring the protocol and port number in packet forwarding. The VM-flow level hashing forces the packets (including the control packets) of one VM-flow to traverse the same path, thus the bandwidth allocation of *eBA* can be accurately enforced. The solution requires a minor modification on switching software, which, however, is able to be implemented in datacenter environment. The coarse-grained ECMP is the price paid to implement the current version of LRC. We will solve this limitation by designing a multi-path LRC is our future work.

VII. Performance Evaluation

We aim at answering two questions in the evaluations: (i) Whether *eBA* can achieve the basic requirements as specified in Sec. I? (ii) How does *eBA* perform under dynamic traffic comparing with other rate control algorithms?

In our testbed, we implement *eBA* by updating the rate factors in Open vSwitch [23], and using Traffic Control (TC) tool to limit the rate of each VM-flow in a Linux based OS. The link capacity is 1Gbps. For α and β, we set $\beta = 2\alpha$. The product $\alpha\tau_r$ is set to 0.5

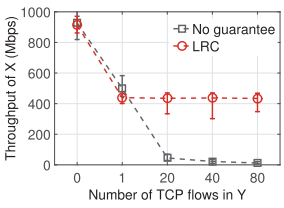


Fig. 8. Throughput of X (450Mbps guarantee) with different number of TCP flows of Y.

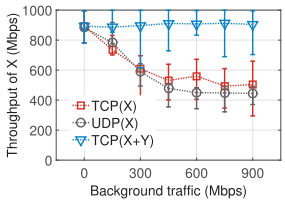


Fig. 9. Throughput of X with increasing throughput of Y (both 450Mbps guarantees).

TABLE III
RATE OF X AND Y WITH DIFFERENT WEIGHT RATIOS

Weight of $w_X : w_Y$	1:1	1:2	1:4	1:8
Throughput of X (Mbps)	441.0	306.9	190.5	112.9
Throughput of Y (Mbps)	444.1	580.2	687.9	747.9
Ratio of $v_X : v_Y$	1:1.01	1:1.89	1:3.61	1:6.62

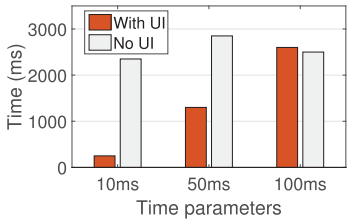


Fig. 10. The average convergence times of long-lived flows under different time parameters. UI shortens the convergence process under smaller time parameters.

A. Achieving the Basic Requirements

The experiments focus on the bandwidth allocation on a bottleneck link, where VM X and Y compete for the limited bandwidth (similar to Fig. 4). The results can be divided into three groups, each targeting at one aspect of the requirements.

Bandwidth Guarantee: In this experiment, X and Y, both with 450Mbps bandwidth guarantee, are sending data to different remote VMs which have the same guarantee. Fig. 8 plots the throughput of X while increasing the number of TCP flows of Y. With no guarantee, the throughput of X decreases sharply since Y has more flows than X. With LRC, the rate of X stays around 450Mbps irrespective of the flow-level competition. This indicates that the traffic of Y is rate limited, and LRC can provide application layer bandwidth guarantees for VMs by limiting other aggressive VM-flows.

Work-Conserving: As LRC guarantees bandwidth for VMs, another question is whether LRC will share the spare bandwidth from underutilized guarantees among unsatisfied VM-flows. The results are shown in Fig. 9. In this example, X with one TCP flow competes with TCP/UDP background traffic of Y. Both X and Y have 450Mbps guarantees, and we plot the throughput of X with an increasing traffic of Y from 0 to 900Mbps. As Fig. 9 shows, when the rate of Y is below 450Mpbs, the rate limit of X is above 450Mbps. The total throughput on the link, which is around 900Mbps, indicates that X utilizes the spared bandwidth and the allocation is work-conserving. Note that X's throughput maintains at about 450Mbps. It validates that LRC guarantees the bandwidth of X in spite of the interference of TCP/UDP background traffic.

Proportional Share: We assign weights to VMs such that the weight ratio of X's VM-flow to Y's VM-flow varies from $1:2$ to $1:8$. Table III presents the throughputs of VM X and Y. When the weights of VM-flows are close to each other (weight ratio < 2), the rates are proportional to their respective weights. However, the ratio of rates deviates from the weight ratio, when the latter ($w_X : w_Y$) grows larger. We trace the logs of received rate factors, and find that the cause is in the measurement error of link throughput in our testbed. The error brought by the received rate factor is non-uniformly magnified by the weights.

B. Algorithm Efficiency

Stable or Fast: Generally, by increasing the frequency of rate control, *eBA*'s rate limit can adapt to the change of traffic faster. Within an acceptable overhead caused by the computation of rate limit and rate factor, it is preferable to use smaller time parameters. However, a key observation from the experiments is that the average rates of links measured by switches become unstable when the time period is too small, e.g., 10ms, due to the congestion control of TCP flows. We now investigate the stability and swiftness of *eBA* in bandwidth allocation, and show how traffic adaptation mechanisms impact the efficiency of algorithms.

Fig. 10 shows the average convergence times of *eBA* under different time parameters. We set up two scenarios similar to that in Fig. 9, where *With UI* means uniform increase is applied in the algorithm, and *No UI* refers to no uniform increase. We observe that reducing the time parameters has little impact on the convergence of the algorithms without uniform increase. This validates our derivation in Sec. III-C that the convergence of the rates is independent of time parameters. In fact, when the time parameters become smaller, the increase/decrease of r and v in each iteration also becomes smaller, as shown in Eq. (15). Consequently, the convergence speed of *eBA* remains unchanged. However, with uniform increase, the change of rate limit and rate factor in each iteration is not reduced by time parameters. When the update of rates becomes more frequent, the convergence becomes faster and the efficiency of the algorithm is also improved.

To show the stability of *eBA*'s rate control, Fig. 11 plots the throughput of VM-to-VM traffic when getting convergent under different time parameters. The mild fluctuation of throughput by static bandwidth reservation indicates that a software rate-limiter can hardly guarantee an absolute stable

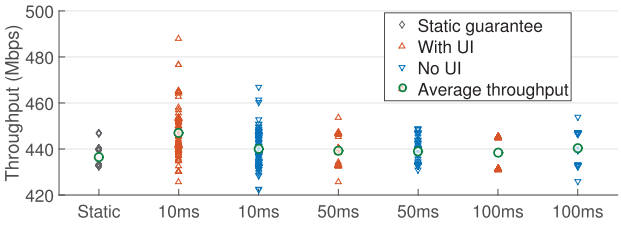


Fig. 11. The variation of throughputs of long-lived flows under different time parameters. As the time parameters decrease, the fluctuations of VM traffic become more prominent. However, the error of through is less than 5% in 98% of duration.

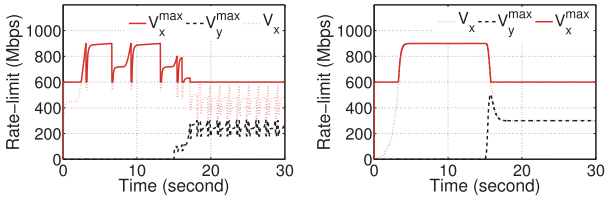


Fig. 12. Rate limit of VMs with $B_X = 600$Mbps, $B_Y = 0$ and the same weight. ($v_X^{max} = \max\{v_X, B_X\}$ and v_X is the rate limit of the algorithms). (a) TCP-like. (b) LRC.

throughput for VM traffic. This is also a challenge for *eBA* to achieve stable bandwidth guarantee using small time parameters. As we can see, with the decrease of time parameters, the fluctuations of VM traffic are more prominent. However, the throughput of VMs is still within an acceptable range of less than 5% error in bandwidth guarantee, which is stable enough for TCP traffic. Besides the variation in rate measurement, we also find that the increase of α, β can magnify the fluctuations, since uniform increase has larger α, β under smaller time parameters. Hence, the choice of time parameters is a tradeoff between fast and stable/overhead. In these tradeoffs, uniform increase enables providers to highly accelerate the convergence for dynamic traffic without sacrificing too much on stability. Also, idle relaxation further reduces the overhead cost by fast convergence.

C. Performance on Dynamic Requirements

To precisely qualify the dynamic of rate limit in bandwidth allocation process, we simulate the algorithms and record the rates of VM-flows within each iteration. We compare LRC with a typical TCP-like rate control algorithm presented in [13], which is based on TPC-CUBIC [24]. Since [13] uses bandwidth guarantees as the weights of VMs, we decouple the guarantee from the weight so as to compare it with LRC. In the simulations, the link capacity is set as 900Mbps, and the rate allocation period is the same for TCP-like algorithm and LRC, both at 15ms. Particularly, we set $\alpha = \beta = 2\text{s}^{-1}$, $\tau_r = 100$ms, $\tau_v = 50$ms, similar to the experimental settings.

Convergence and Congestions: Firstly, we characterize the convergence process under congestions on a single bottleneck.

In the first case, VM X has a guarantee of 600Mbps. When $t = 15$s, a VM-flow (no guarantee) from Y joins in. Fig. 12 shows the dynamic of rate limit from 0 to 30s. Both LRC and the TCP-like algorithms have fast convergence speed to

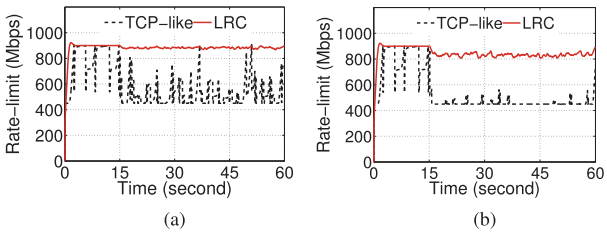


Fig. 13. Rate limit of X (450Mbps guarantee) with arrival of short flows at 15^{th}s: 5 flows/s at average 17.4Mbps and 20 flows/s at average 66.2Mbps. (a) 5 short flows per sec. (b) 20 short flows per sec.

high utilization (4s) and fairness (3s). LRC is smooth during the whole period. However, in the TCP-like algorithm, when the VM-flow from Y joins in, the rate limit for X becomes fluctuating until it gets below the guarantee, and the rate limit for Y keeps fluctuating. The fluctuation leads to frequent changes of the rate limit on underlying TCP flows, and may deteriorate the performance of the transport layer.

One may question why the rate limit in the TCP-like algorithm becomes more fluctuating when Y joins in. The reason is that TCP-like algorithms use multiplicative decrease when congestion occurs. In addition, in [13], the increment of rate has an inverse correlation with the rate (the authors call it as *rate-caution*). As a result, the VM-flow with small rate using rate-caution will be more aggressive than that using the original TPC-CUBIC protocol, thus fluctuations are more frequent when Y joins in. On the contrary, our algorithm applies a smooth rate limit to VMs, and will not decline the rate of TCP flows.

Frequent Short Flows: In this simulation, we evaluate the algorithms under frequent short flows. We consider the situation where VM X is sending long flows with a guarantee of 450Mbps, and has fully utilized the bottleneck link. We generate two groups of short flows passing through the bottleneck link of X, where one group with 5 flows/s and the other with 20 flows/s. The short flows are generated uniformly at random in each second from 15^{th}s to 60^{th}s, and the flow sizes are exponentially distributed around a mean of 200KB. Fig. 13 shows the rate limit of X with LRC and the TCP-like algorithm. The TCP-like algorithm suffers severe drops in rate limit. The short flows create bursty traffic (17.4Mbps for 5 flows/s and 66.2Mbps for 20 flows/s, measured by 15ms) along with existing flows, leading to a waste of network bandwidth since the TCP-like algorithm enforces fluctuant rate limit for the VM-flows. However, our algorithm is less sensitive to short flows. Even with 20 flows per seconds, the rate limit can stay flat and the decrement is about the rate of all short flows. Such observations validate that LRC has advantages over the TCP-like algorithm under frequent short flows. Hence, we believe LRC is more suitable for datacenter environment.

Performance With Mapreduce Workloads: To verify the performance of LRC under datacenter traffic, we develop a simulator to model VM bandwidth sharing in datacenters. The simulator simulates a two-level network topology with 600 servers. Each server hosts 4 VMs and has 1Gbps bandwidth. 40 servers form a rack connected by a

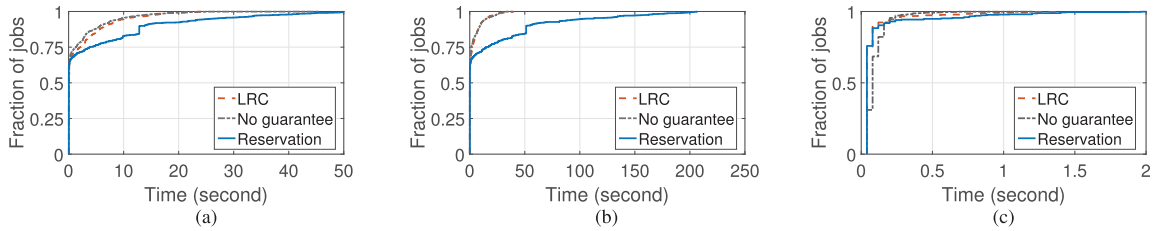


Fig. 14. The CDF of shuffle time for LRC, reservation and best effort manner. LRC achieves 96% network utilization of the best effort manner. LRC's performance on jobs with short shuffle size approximates the reservation policy and outperforms the reservation policy in jobs with large shuffle size. (a) Full bisection bandwidth. (b) $4\times$ oversubscription. (c) 600 small jobs.

Top-of-Rack (ToR) switch. 15 ToR switches are linked to the root switch. We set up two scenarios: (i) full bisection bandwidth, where the uplink bandwidth of ToR switch is 40Gbps, and (ii) $4\times$ oversubscription bandwidth [9], where the uplink bandwidth of ToR switch is 10Gbps. We simulate Mapreduce jobs running in the cluster. Each job needs to transmit data from map nodes (VMs) to reduce nodes (VMs) in the shuffle phase. The simulated Mapreduce workloads are collected from the same number of servers in Facebook datacenter [7], consisting of up to 900 jobs. The maximum transmitted data size is 32.3GB and the minimum is 0. The bandwidth guarantees for each VM in LRC and reservation policy are set to 250Mbps and 62.5Mbps, for full bisection bandwidth and $4\times$ oversubscription, respectively.

Fig. 14 shows the CDF of completion time of shuffle phase (shuffle time). About 600 jobs have small shuffle size less than 128MB, among which 140 jobs are very short with $<$ 1MB shuffle size. Their shuffle times are short (about $<$ 1s) and the workloads are less network-sensitive. As shown in Fig. 14(c), LRC performs approximately to the reservation policy and does not incur high delays for these jobs. The shorter completion time of *eBA* shows *eBA*'s bandwidth guarantee can protect the network performance of these jobs as compared with the best-effort manner. For jobs with large size of data to shuffle, the shuffle phase in LRC is faster than that of the reservation policy and the speedup is more obvious in $4\times$ oversubscription networks. This fits in with our expectation, since LRC ensures a lower bound bandwidth equivalent to the reserved bandwidth, and utilizes the spared bandwidth at the same time. In comparison with the best effort manner, LRC is a bit slower for large jobs under full bisection bandwidth. The drawback of LRC for large jobs comes from that LRC makes the underlying TCP flows less aggressive when the rate approaches the maximum. However, LRC still achieves 96% performance of the best effort manner, which validates the high network utilization of LRC.

VIII. Related Work

Towards achieving predictable network performance for cloud applications, researchers have proposed numbers of approaches to share bandwidth in datacenter networks. The main ideas in these works are two fold: The first idea focuses on VM allocation in datacenters, such as [9] and [11]. Oktopus [9] uses VM placement to provide bandwidth guarantees. Silo [25] takes a step forward to achieve latency guarantee by using traffic pacing on rate control. They both enforce static rate limits to reserve bandwidth for VMs. While these policies focus on predicable performance for VMs, they ignore the dynamic feature of datacenter traffic. Proteus [11] proposes a time varying reservation policy based on the bandwidth requirements of specific Mapreduce applications. However, the solution is limited to a few application types.

The other idea for sharing datacenter network is to allocate bandwidth for VMs after their placement by enforcing dynamic rate limit. Faircloud [5] presents the basic bandwidth requirements of bandwidth allocation problem and proposes three kinds of sharing policies. NetShare [26] achieves proportional bandwidth sharing among different VMs by using weighted fair queues. The policies in the above proposals need the support of per-VM queue in switches for rate control. This means they are hard to be scaled, due to the limited queues supported by each port at switches. The configuration is also complicated, as the communication patterns among VMs are changing. References [10], [12], and [13] leverage end-based rate limit to achieve work-conserving bandwidth allocation for VMs. Seawall [10] and ElasticSwitch [13] use TCP-CUBIC based rate control, and have fluctuations under bursty traffic when the rate limit is beyond the guarantee. ElasticSwitch also needs to number the packets for each destination in the switch, which is more complicated than our solution. EyeQ [12] uses a variant of RCP. EyeQ assumes a congestion free core and the rate control algorithm only has a linear convergence speed. References [17] and [27] develops a game theoretical allocation strategy that can flexibly balance the guarantee-proportionality tradeoff. The main drawback is its relying on precise traffic demand prediction. Finally, [28] applies the Logistic model in congestion control algorithms at the transport layer in the Internet, while our work use it for rate limits of VMs.

IX. Conclusion

Traffic in data centers are highly dynamic and bursty due to massive number of short flows. Previous work on bandwidth allocation for datacenters are not appropriate since rate limiting traffic is ineffective and causes system performance degradation. In this paper, we presented *eBA*, a distributed solution for bandwidth guarantee, using the Logistic model under the control-theoretic framework. Unlike previous proposals, our solution provides a stable and fast-convergent allocation process, which meets the basic and dynamic requirements of resource sharing in datacenter networks. *eBA*'s can be deployed in datacenters with no changes to network protocol

stack or switching hardware, and shows effectiveness in coping with traffic variability in datacenters, hence giving public cloud providers an additional performance guarantee feature to users.

REFERENCES

[1] *Amazon Elastic Compute Cloud*, accessed on 2015. [Online]. Available: http://aws.amazon.com

[2] F. Xu *et al.*, "iAware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3012–3025, Dec. 2013.

[3] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proc. IEEE*, vol. 102, no. 1, pp. 11–31, Jan. 2013.

[4] Z. Zhou *et al.*, "On arbitrating the power-performance tradeoff in SaaS clouds," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 872–880.

[5] L. Popa *et al.*, "FairCloud: Sharing the network in cloud computing," in *Proc. ACM SIGCOMM*, 2012, pp. 187–198.

[6] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. ACM IMC*, 2009, pp. 202–208.

[7] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads," *Proc. VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.

[8] J. Lee *et al.*, "Application-driven bandwidth guarantees in datacenters," in *Proc. ACM SIGCOMM*, 2014, pp. 467–478.

[9] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. ACM SIGCOMM*, 2011, pp. 242–253.

[10] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proc. USENIX NSDI*, 2011, pp. 309–322.

[11] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: Incorporating time-varying network reservations in data centers," in *Proc. ACM SIGCOMM*, 2012, pp. 199–210.

[12] V. Jeyakumar *et al.*, "EyeQ: Practical network performance isolation at the edge," in *Proc. USENIX NSDI*, 2013, pp. 297–312.

[13] L. Popa *et al.*, "ElasticSwitch: Practical work-conserving bandwidth guarantees for cloud computing," in *Proc. ACM SIGCOMM*, 2013, pp. 351–362.

[14] J. Guo *et al.*, "On efficient bandwidth allocation for traffic variability in datacenters," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1572–1580.

[15] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[16] J. Guo, F. Liu, J. C. S. Lui, and H. Jin, "Fair network bandwidth allocation in IaaS datacenters via a cooperative game approach," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 873–886, Apr. 2015.

[17] J. Guo, F. Liu, D. Zeng, J. C. S. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2139–2147.

[18] J. D. Murray, *Mathematical Biology*. New York, NY, USA: Springer, 2002.

[19] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, vol. 60. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[20] D. P. Bertsekas and R. G. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1992.

[21] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2007.

[22] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. USENIX NSDI*, 2010, p. 19.

[23] *Open vSwitch: Production Quality, Multilayer Open Virtual Switch*, accessed on 2015. [Online]. Available: http://openvswitch.org/

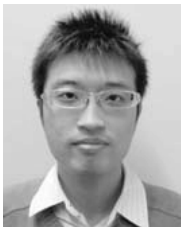
[24] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[25] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable message latency in the cloud," in *Proc. ACM SIGCOMM*, 2015, pp. 435–448.

[26] T. Lam and G. Varghese, "NetShare: Virtualizing bandwidth within the cloud," Dept. Comput. Sci. Eng., UCSD, La Jolla, CA, USA, Tech. Rep., 2009.

[27] J. Guo *et al.*, "Falloc: Fair network bandwidth allocation in IaaS datacenters via a bargaining game approach," in *Proc. IEEE ICNP*, Oct. 2013, pp. 1–10.

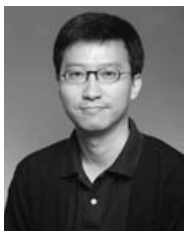
[28] X. Huang *et al.*, "Improving the convergence and stability of congestion control algorithm," in *Proc. IEEE ICNP*, Oct. 2007, pp. 206–215.



Fangming Liu (S'08–M'11–SM'16) received the B.Eng. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2005, and the Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2011. He is currently a Full Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and datacenter, mobile cloud, green computing, SDN/NFV, and virtualization. He is selected into National Program for Support of Top-Notch Young Professionals of National Program for Special Support of Eminent Professionals. He is a Youth Scientist of National 973 Basic Research Program Project of SDN-Based Cloud Datacenter Networks. He was a StarTrack Visiting Faculty Member with Microsoft Research Asia from 2012 to 2013. He has been the Editor-in-Chief of *EAI Endorsed Transactions on Collaborative Computing* and a Guest Editor of the *IEEE Network Magazine*, and served on the TPC of ACM Multimedia 2014 and 2016, e-Energy 2016, the IEEE INFOCOM 2013-2017, ICNP 2014 TPC, and 2016 Poster/Demo Co-Chair of IWQoS 2016-2017 and ICDCS 2015-2016. He is a Senior Member of the IEEE.

Jian Guo received the B.S. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, China, where he is currently pursuing the Ph.D. degree. His research interests include data center networking and software-defined networking.

Xiaomeng Huang received the B.A. degree from Wuhan University, Wuhan, China, in 2000, the M.S. degree from the Huazhong University of Science and Technology, Wuhan, in 2003, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2007. He is especially interested in parallel computing and distributed system.



John C. S. Lui (A'92–M'93–SM'02–F'10) received the Ph.D. degree in computer science from the University of California at Los Angeles, CA, USA, in 1992.

He is currently a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (CUHK), Hong Kong. He was the Chairman of the department from 2005 to 2011. His current research interests are in communication networks, network/system security (e.g., cloud security and mobile security), network economics, network sciences (e.g., online social networks and information spreading), cloud computing, large-scale distributed systems, and performance evaluation theory.

Prof. Lui is a Fellow of the Association for Computing Machinery (ACM), a Croucher Senior Research Fellow, and an Elected Member of the IFIP WG 7.3. He received various departmental teaching awards and the CUHK Vice Chancellor's Exemplary Teaching Award. He was a co-recipient of the IFIP WG 7.3 Performance 2005 and the IEEE/IFIP NOMS 2006 Best Student Paper awards. He serves on the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the *Journal of Performance Evaluation*, and the *International Journal of Network Security*.