Pricing Intra-Datacenter Networks with Over-Committed Bandwidth Guarantee

Jian Guo, Fangming Liu, and Tao Wang, ***Key Laboratory of Services Computing Technology and System, Ministry of Education, School of Computer Science and Technology, Huazhong University of Science and Technology;*** **John C.S. Lui,** ***The Chinese University of Hong Kong***

https://www.usenix.org/conference/atc17/technical-sessions/presentation/guo-jian

This paper is included in the Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC '17).

July 12–14, 2017 • Santa Clara, CA, USA

ISBN 978-1-931971-38-6

Open access to the Proceedings of the 2017 USENIX Annual Technical Conference is sponsored by USENIX.

Pricing Intra-Datacenter Networks with Over-Committed Bandwidth Guarantee

Jian Guo[1], Fangming Liu[1]*, Tao Wang[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*
[2]*The Chinese University of Hong Kong*

Abstract

Current IaaS clouds provide performance guarantee on CPU and memory but no quantitative network performance for VM instances. Our measurements from three production IaaS clouds show that for the VMs with same CPU and memory, or similar pricing, the difference in bandwidth performance can be as much as 16×, which reveals a severe price-performance anomaly due to a lack of pricing for bandwidth guarantee. Considering the low network utilization in cloud-scale datacenters, we address this by presenting SoftBW, a system that enables pricing bandwidth with over commitment on bandwidth guarantee. SoftBW leverages usage-based charging to guarantee price-performance consistency among tenants, and implements a fulfillment based scheduling to provide bandwidth/fairness guarantee under bandwidth over commitment. Both testbed experiments and large-scale simulation results validate SoftBW's ability of providing efficient bandwidth guarantee, and show that by using bandwidth over commitment, SoftBW increases 3.9× network utilization while incurring less than 5% guarantee failure.

1 Introduction

Cloud computing enables enterprises and individuals to access computing resources based on their demands with a simple pay-as-you-go model. Large numbers of cloud tenants are multiplexed in the same datacenter (DC) infrastructure, where they can also get isolated computing resources via virtual machines (VMs). In current IaaS clouds, CPU performance (represented by vCPUs) and memory performance (represented by GB) are both quantifiable metrics. However, datacenter network bandwidth, which can severely impact the job completion time of network-intensive applications, has not been standardized as one of the VM performance metrics.

To reveal the network performance in IaaS clouds, we measured the intra-datacenter bandwidth of VMs from three popular cloud platforms, i.e., Google Compute Engine (GCE), Amazon EC2 and Aliyun ECS. The measurement indicates a severe *price-performance anomaly* in current clouds: 1) for VMs from different clouds, whose CPU/memory and prices are the same, the difference in network performance can be as much as 16 times; 2) for VMs in the same cloud, the average bandwidth of a cheaper VM can surpass the bandwidth of an expensive one; 3) for a single VM, the network performance at different time is varying and highly unpredictable. Hence, due to a lack of bandwidth performance guarantee and a corresponding pricing strategy, tenants deploying network intensive applications can hardly obtain the performance in agreement with which they pay for.

To price network bandwidth with a performance guarantee, a practical solution should not only satisfy the bandwidth requirements of tenants, but should also achieve efficient resource utilization to benefit providers' profit. Given that current cloud-scale datacenters have a low network utilization (99% links are less than $<$ 10% loaded [1]), we propose to allow over commitment for bandwidth guarantee as well as provide usage-based pricing for tenants. For example, a provider can co-locate VM_1 with 8 Gbps bandwidth and VM_2 with 4 Gbps bandwidth on a server with 10 Gbps bandwidth. When both VMs are transmitting traffic continuously, VM_1 and VM_2 get 6.7 Gbps and 3.3 Gbps bandwidth, respectively. For a given billing cycle, they both get a discount in proportional to the fulfillment ratio, i.e., $5/6$. To validate the feasibility of bandwidth over commitment, we model the failure rate of bandwidth guarantee based on the datacenter traffic traces in [2, 3], and show that we can control

This work was supported in part by the National 973 Basic Research Program under Grant 2014CB347800, and in part by NSFC under Grant 61520106005. The work of John C.S. Lui is supported in part by the GRF 14205114. (*Corresponding author: Fangming Liu)

the expected failure rate within an acceptable level by using proper over commitment ratio (§2).

However, pricing bandwidth guarantee under over commitment condition is a challenging task. Existing work on bandwidth guarantee, which focuses on guaranteeing tenants' bandwidth requirements under sufficient bandwidth condition, can hardly achieve the target:

- They do not provide a usage-based pricing for the corresponding bandwidth allocation techniques to achieve price-performance consistency.
- The state-of-the-art bandwidth allocation solution using either static or dynamic rate limit cannot provide performance guarantee under bandwidth over commitment.
- Existing solution verifies bandwidth guarantee by using long-lived flows, but ignores the performance degradation of short flows when using periodically rate limit.

To address the challenges, we propose SoftBW, a solution that enables pricing datacenter networks via software-defined bandwidth allocation, aiming to achieve: price-performance consistency, over commitment tolerance and short flow friendly (§4). SoftBW realizes a usage-based charging by monitoring a guarantee fulfillment, which is a ratio of the achieved bandwidth to the committed bandwidth guarantee, on each billing cycle. The pricing strategy, when combined with our bandwidth allocation, ensures that tenants paying higher unit price can achieve higher network performance.

SoftBW implements a fulfillment-based scheduling (§5) to simultaneously provide minimum bandwidth guarantee under sufficient physical bandwidth condition, and guarantee VM-level fairness when the physical bandwidth is constrained. By applying a dynamic guarantee for long-lived traffic, SoftBW can utilize the idle bandwidth to reduce the total bandwidth guarantee, which further reduces the guarantee failures under bandwidth over commitment. SoftBW's implementation uses a software virtual switch at each server, and introduces only 5.1% CPU overhead and less than 1.9 μs latency for 10 Gbps data transmission. Testbed experiments validate SoftBW's ability to efficiently provide bandwidth guarantee, as well as having 2.8$\times$ to 4.5$\times$ improvement on the completion time of short flows, as compared with existing rate-limit based approaches. In large scale simulation, we find that using bandwidth over commitment can increase $\sim$ 3.9$\times$ network utilization, while maintain the average failure of bandwidth guarantee under 5%. In summary, the contributions of this paper consist of:

- By measuring the intra-DC bandwidth of VMs from three cloud platforms, we reveal the severe price-performance anomaly among different clouds, due to a lack of pricing for quantitative bandwidth performance.
- We validate the feasibility of pricing bandwidth guarantees with over commitment in current multi-tenant datacenters by proposing a usage-based charging model and a fulfillment-based scheduling algorithm.
- We develop SoftBW, a system that implements the proposed pricing and scheduling, and show that SoftBW can provide efficient bandwidth/fairness guarantee for both long-lived traffic and short flows under bandwidth over commitment in testbed experiments.

2 Background and Motivation

2.1 VM Bandwidth in Public Cloud

We measure the intra-DC bandwidth among different instance types[1] from four selected datacenters: N. Virginia in US East (Amazon), N. California in US West (Amazon), Asia East (Google), Beijing in China (Alibaba). Each throughput is collected 12 times a day, lasting 5 minutes on every 2 hours. The maximum and average throughput of different instances is shown in Table 1. During the measurement, the CPU utilizations of all VMs are less than 100% of a single core, which indicates that the bottleneck is on the network bandwidth. The instance types with the same network performance are merged into one group. We find that while VM rate-limit is commonly used, the *limited bandwidth* and *direction of rate-limit* for VMs are quite different among different providers.

Rate-limit upperbound. Alibaba ECS simply provides the same rate-limit for all VMs at 520 Mbps. For Amazon EC2, the sharing strategy of the two datacenters sees no obvious distinctions. As expected, the performance corresponds to the description of VM instances (excluding the ones with 10 Gbps dedicated bandwidth) in EC2, which falls into three levels, i.e., low to moderate, moderate (300 Mbps) and high (1 Gbps). The throughput of "low to moderate" is fairly unstable and can be as large as about 3 Gbps. Although Google does not claim the network performance of VMs, most of them outperform the VMs in EC2 in both average and maximum throughput, which has three levels, i.e., 1 Gbps, 2 Gbps and 4 Gbps. For VMs with 4 or more vCPUs, we do not find any obvious rate-limit. The stable throughput varies from 1 Gbps up to 5 Gbps.

Rate-limit direction. There are two rate-limit strategies in these clouds: source based rate-limit for egress traffic (Google and Alibaba), and rate-limit for both

[1]The measurement covers all shared-core and standard instances in GCE (cloud.google.com), all general purpose instances in EC2 (aws.amazon.com), and all #vCPUs in Alibaba ECS.

EC2	Low to Moderate	Moderate	High
Low to Moderate	0.84/3.00	0.29/0.30	0.79/1.01
Moderate	0.29/0.30	0.29/0.29	0.29/0.29
High	0.55/1.01	0.29/0.30	0.97/1.16

GCE	Low	Moderate	High	Highest
Low	0.59/1.01	0.66/1.01	0.64/1.01	0.64/1.00
Moderate	0.76/2.00	1.97/2.00	1.99/2.00	1.98/2.00
High	0.65/3.36	2.78/3.24	2.73/3.20	3.03/4.00
Highest	0.86/4.93	3.32/4.05	3.56/3.95	4.36/5.09

Table 1: The average/max throughput of VM-to-VM traffic (Gbps) in Amazon EC2 and Google Compute Engine (GCE) datacenters. The left column and head row are source and destination VM, respectively. [1]

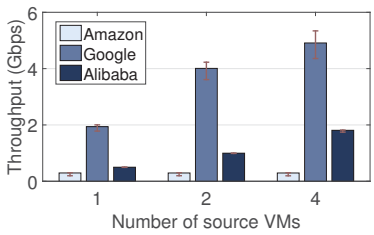

Figure 1: Comparing the average, max, min receiving rate of VM with different numbers of sending VMs.

ingress and egress traffic (Amazon). In Figure 1, we validate this by showing the receiving rates of VMs with different number of sending VMs. The instance types we studied in three clouds are guaranteed to have the nearest performance in CPU and memory: 1 vCPU, 3.75 GB memory for GCE, 1 vCPU, 3.75 GB memory for EC2, and 1 vCPU, 4 GB memory for ECS. The receiving rates of VMs in EC2 do not increase with more sending VMs, which indicates that the ingress bandwidth of VM is limited. GCE and ECS have no rate-limit for ingress traffic of VMs unless they are congested by physical bandwidth. Hence, the maximal rates can reach at about 5 Gbps and 2 Gbps with 4 sending VMs, respectively.

Price-performance anomaly. For VMs in different clouds, which have the same allocated resource (i.e., vCPUs and memory) and pricing, the bandwidth performance is significantly different. For example, the VMs in Figure 1 have 1 vCPU and 3.75 GB memory, but the gap in network performance is as much as 6 to 16 times. As a result, some cloud providers may miss a golden opportunity to achieve higher competitiveness in the market due to a lack of quantitative bandwidth performance. In fact, as indicated by the missing of bandwidth performance for VMs, currently the maximal throughput is not guaranteed by the provider since we observe significant variation in throughput during one day.

2.2 Why Over Commitment is Rational?

To validate the feasibility of over selling network bandwidth in clouds, we start with modeling the datacenter traffic based on existing measurement work [2, 3]. The ratio of over commitment of a server is defined as the ratio of the sum of guaranteed bandwidth C_B to the physical bandwidth C at this server, denoted by $\delta = C_B/C$. When over commitment is involved, the risk of guarantee failure caused by resource over commitment should be considered.

Suppose the server hosts n homogeneous VMs with the same bandwidth guarantee, whose traffic is independent identically distributed. As indicated by [2], the traffic demand on an edge port approximates an exponential distribution with the probability density satisfying $f(x) = \alpha e^{-\alpha x}$, where x is the traffic demand and $1/\alpha$ is the average traffic demand. Note that $1/\alpha$ can be obtained by tracking the average network utilization of a VM. In a δ over committed server, where the bandwidth guarantees of VMs exceed the physical bandwidth, the guarantee fails when the total traffic demands exceed the physical bandwidth C, namely, $\sum x_i > C$, where x_i represents the demand of VM i, $i \in [1,n]$.

Let Θ denote the domain that subjects to $\sum x_i > C (x_i > 0)$ for those n VMs. Then the probability of failure P_n follows the joint probability distribution that every x_i locates in Θ, which is an n-dimensional integral

$$P_n = \int \dots \int_{\Theta} \prod \alpha e^{-\alpha x_i} \mathrm{d}x_1 \dots \mathrm{d}x_n. \quad (1)$$

Solving above equation, P_n can be expressed as

$$P_n = e^{-\alpha C} \sum_{i=1}^{n} \frac{(\alpha C)^{i-1}}{(i-1)!}. \quad (2)$$

Let ρ be the average network utilization of a host server without over commitment, then $\alpha = n/\rho C_B$. Figure 2 shows the impact of over commitment on the failure rate with 16 VMs. We maintain the server network utilization as 10% and 15% (the value can be adjusted according to the network utilization by providers). As we can see, the failure rate is less than 5% if using 6.9× OC for 10% average network utilization, or using 4.6× OC for 15% average network utilization.

The simple analysis shows that when the access bandwidth is over committed, the expected failure rate can be controlled within an acceptable level by using proper ratio of over commitment according to the average utilization. Although in practical situation, a VM's traffic may not follow an ideal exponential distribution, the over commitment is still worth deployment in a large scale,

[1] There is no description for VM network performance in GCE. The measured performance can be divided into 4 groups: low (f1-micro, g1-small), moderate (n1-standard-1), high (n1-standard-2), and highest (n1-standard-4/8/16)). The inter-VM throughput of Alibaba is omitted as the strategy is similar to GCE.

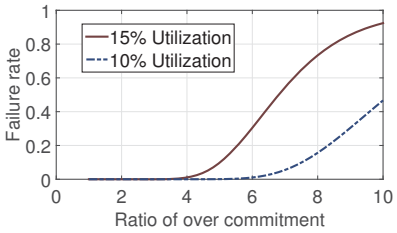

Figure 2: The impact of bandwidth over commitment on the failure rate of bandwidth guarantee.

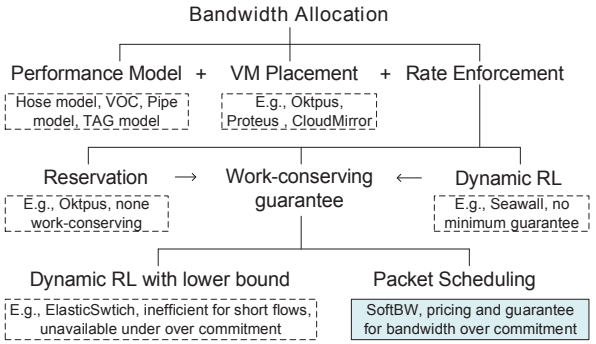

Figure 3: Technical position of SoftBW in bandwidth allocation: rate enforcement with packet scheduling.

and the VMs still get a minimum guarantee of C_B/δ in the worst case.

2.3 Why not Existing Solutions?

Achieving bandwidth guarantee for VMs in datacenters needs to address three key tasks [4] as shown in Figure 3: a performance model that specifies the tenants' bandwidth requirements, a VM placement mechanism that allocates VMs to the servers with sufficient physical bandwidth, and a rate control algorithm that dynamically controls the rates of VMs to improve bandwidth utilization. Current rate-limit (RL) based solution can meet the basic requirements of bandwidth guarantee [5, 6], i.e., minimum guarantee, proportional sharing, and work-conserving. However, they do not provide a pricing strategy and can hardly address the challenges on bandwidth over commitment in datacenters.

First, the rate-limit based solution does not provide guaranteed performance metrics for bandwidth over commitment. They work like TCP for aggregated VM-to-VM traffic, i.e., keep increasing and multiplicatively decrease when congested, so as to provide VM-level fairness. To achieve bandwidth guarantee for a VM, the rate limitation needs to stay above the minimum guarantee, thus the limitation of traffic from other VMs will reduce, and their traffic that exceeds the guaranteed bandwidth can be restricted [5]. This policy assumes that the access links at end-hosts are not over-subscribed. When the total bandwidth guarantee exceeds the physical bandwidth on a server, for example, three VMs each with 500 Mbps minimum bandwidth guarantee are co-located on a server with 1 Gbps bandwidth, the minimum rate-limit of each VM (500 Mbps) is held upon a fair share (333 Mbps), thus becoming unavailable. To avoid this, one should tell whether the total traffic demand will exceed the physical bandwidth in the next update of rate limitations, and decide whether the limitations need to be lower-bounded. However, predicting traffic demands at $\sim$ 50 ms granularity is extremely hard for datacenter traffic. An efficient rate enforcement mechanism is needed to guarantee fairness when the physical bandwidth is over committed while providing minimum bandwidth under sufficient bandwidth condition.

Second, periodically rate limit degrades the performance of VM-to-VM traffic when the traffic is an aggregation of massive short flows. To achieve work-conserving [7], the unused bandwidth of idle VMs is allocated to other VMs rather than be statically reserved. Hence, the rate limitations of VMs need to be updated at an interval of tens of milliseconds [5], which is longer than the completion times of most of short flows. During this period, if the traffic of a VM has fully utilized the allocated bandwidth, the newly arrived short flows will compete with existing traffic and cause a short-term congestion. It not only delays the transmission of short flows, but also degrades the performance of existing flows. One may address this by using more fine-grained rate control. However, frequent changes of rate limitation, especially sudden decrease in rate limitation, can cause significant fluctuations to the underlying TCP flows. For a tenant, it is unacceptable that the use of idle bandwidth is at the cost of degrading the performance of short flows. Hence, a packet-level solution that can "take back" the paid bandwidth quickly is more suitable for bandwidth pricing.

3 Fulfillment Abstraction

3.1 Access Bandwidth vs. Congested Links

Our bandwidth allocation focuses on end-based rate enforcement, as shown in Figure 3. The choice comes from the fact that today's production datacenters see rapid advances in achieving full bisection bandwidth [8, 9], and the providers have a growing concern about the over committed access bandwidth on each server rather than the aggregation and core level. By leveraging the software virtual switch at each server, the cost of implementation can be reduced and the scale of rate control is limited to the number of VMs on each server. Our design as-

sumes the datacenter fabric to be a non-blocking switch [10, 11, 7], and our main focus is to schedule the traffic at the virtual ports connected to VMs.

3.2 Guarantee Fulfillment

Our work aims at enforcing the bandwidth at the VM-level, and providing pricing schemes for bandwidth guarantee. This requires an abstraction that not only provides a performance metric for bandwidth sharing under bandwidth over commitment, but also serves as a quota for charging. To this end, we propose the concept of *guarantee fulfillment* to express tenants' bandwidth performance. The fulfillment is defined as the ratio of VM x's rate r_x to its promised bandwidth guarantee B_x:

$$F_x = \frac{r_x}{B_x}. \quad (3)$$

As the fulfillment takes a bandwidth guarantee as a baseline, it is complementary to existing network model for expressing tenants' bandwidth requirements. For example, B_x can be the VM bandwidth in a Virtual Cluster [10] model. We define the bandwidth guarantee for each VM since it can be better adapted to current per-VM based charging in cloud. Note that the abstraction can also be extended to the VM-to-VM bandwidth in a Tenant Application Graph model, if we setup a virtual queue for each VM-to-VM pair at both source and destination servers.

Fulfillment for scheduling. The performance guarantee for the tenants relies on maintaining fairness among VMs' fulfillments. When bandwidth is sufficient, the fairness of fulfillments means that for any VM x, y, $F_x = F_y > 1$, and the VMs have minimum bandwidth guarantee since $r_x > B_x$. If the bandwidth is over committed, the VMs may have $F_x < 1$ when the total traffic demand exceeds the physical bandwidth. By maintaining the same fulfillment, network proportionality can be achieved, i.e., $r_x : r_y = B_x : B_y$ for any VM x, y. Thus the worst case performance under δ over commitment will be no less than B_x/δ and B_y/δ.

Fulfillment for pricing. As a charging quota for providers, the fulfillment indicates how much of the paid bandwidth is actually obtained by a tenant. The bandwidth is charged according to the fulfillment of VMs measured on a billing cycle (e.g., per second), where a discount is applied based on the actual usage. The billing cycle is similar to the minimum period for charging in current clouds, e.g., GCE use per-minute billing for VM instances. To price the bandwidth under over commitment, two tasks should be done: First, a model to estimate the failure of guarantee as a service commitment for the failure rate in the SLA (similar to the monthly uptime percentage in EC2 SLA [12]) (§2.2). Second, a *fulfillment-based pricing function* that guarantees tenants paying higher prices can achieve higher performance.

4 SoftBW Design

SoftBW enables pricing intra-DC bandwidth under over commitment by scheduling packets to satisfy VMs' bandwidth requirements and charging based on the actual bandwidth used by VMs. Specifically, our design targets at the following goals:

- **Price-performance consistency.** Tenants paying higher price should achieve proportionally higher bandwidth performance. Tenants can not achieve higher performance by lying about their bandwidth requirements.
- **Over commitment tolerance.** The system should simultaneously provide minimum bandwidth guarantee when physical bandwidth is sufficient, and guarantee fairness when their minimum guarantees exceed the physical bandwidth.
- **Short flow friendly.** The performance of short flows with bandwidth guarantee should not be degraded when the physical bandwidth is occupied by other traffic.

4.1 Architectural Overview

SoftBW uses a Software-Defined Networking architecture where each host server deploys a virtual switch that can be controlled by centralized controllers. As Figure 4 shows, SoftBW leverages a centralized master to manage the business in the control plane, and enforces bandwidth allocation using distributed agents in the data plane. The system consists of two functions: (i) pricing the bandwidth based on the measured fulfillment from the traffic monitor (§4.3), and (ii) enforcing the requirements of bandwidth guarantee by using packet scheduling in the virtual switch (§5). The two functions both have decoupled modules in SoftBW master and SoftBW agent nodes.

SoftBW master maintains the requirements (bandwidth paid by tenants) and fulfillments of VMs at a logically centralized server. The information is used by cloud providers to define their charging models. SoftBW agent leverages the virtual SDN switch at each server to schedule packets from per-VM queues by obtaining the requirements from the controller. The scheduling algorithm works in a round robin manner where the VMs with less fulfillment can be preferentially scheduled. The overhead of virtual SDN switch, which is determined by the number of VMs on each server, does not increase as we scale up the number of servers in datacenters.

SoftBW works as follows. When a VM is launched and connects to a port of the virtual switch, the data plane generates an asynchronous message to the control plane, which notifies the connection of a node. SoftBW master can capture the port connected to the VM, and sends

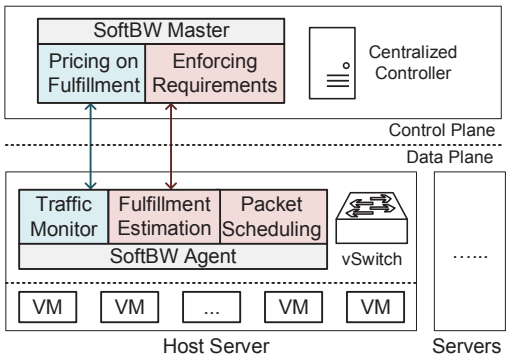

Figure 4: System overview: the data plane schedules packets to satisfy the VMs' bandwidth requirements, and the control plane manages pricing using measured fulfillments.

out the corresponding bandwidth guarantee to the agent. Afterwards, the agent creates a queue for this VM and adds a flow table entry to match and enqueue the packets of the VM. In the lifetime of the VM, the traffic monitor in agent monitors the rates of VMs, and feeds the rates back periodically to compute the price and update the guarantee values of VMs which require the dynamic bandwidth guarantee. Note that the fulfillment for pricing, which is measured at a billing cycle, is isolated from the fulfillment for scheduling at the packet level.

4.2 Performance Metrics

SoftBW allows the provider to take advantage of low network utilization in datacenters to oversell the physical network bandwidth. One important note is that different applications need different kinds of guarantees. For example, delay-tolerant applications like background backup, whose completion times are only related to the total throughput during the period of backup. Therefore, a cloud provider does not need to provide a strict rate guarantee for the entire duration of the backup job. Instead, the bandwidth can be allocated to other VMs which are running real-time jobs, and then compensate the backup jobs when more bandwidth is available. As a result, applications which require strict rate guarantee and applications which require deadline guarantee can both be satisfied.

Differentiated performance metrics. We now propose three different network performances, which allow the bandwidth guarantee can be dynamically allocated, thus to reduce the risk of guarantee failure as well as increase the network utilization under over commitment situation.

- *Strict guarantee* provides the real-time minimum bandwidth guarantee for a VM, which is denoted by a dedicated rate B.
- *Dynamic guarantee* ensures the total deliverable traffic of a VM during a specific time period (e.g., time to deadline). The dynamic guarantee is denoted by a tuple (S,T), where S is the total traffic size and T is the desired transmission time.
- *Fairness guarantee* offers fair VM-level fairness for sharing the residual bandwidth, which is left by VMs with strict or dynamic guarantees, among all VMs.

Note that dynamic guarantee can be satisfied by an average rate of $b' = S/T$ Mbps. However, instead of guaranteeing this average rate for the whole duration T, we vary the bandwidth guarantee according to the traffic loads in datacenters. We first assign an initial minimum guarantee S/T for the VM. After a period of t_0 seconds, there might be s' Mb remaining data on that VM, which should be transmitted in $t' = T - t_0$ seconds. Then we update the guarantee b' as s'/t' Mbps (which is called the expected guarantee), and periodically repeat this update. As a result, the guarantee is dynamically adjusted according to the available bandwidth. If there is residual bandwidth on the server, the VM can utilize it and reduce the guarantee in the next update. As a result, the total bandwidth guarantee on a server is reduced, and the probability of guarantee failure also decreases. If the bandwidth is not guaranteed for some periods, the VM can increase the guarantee and still finish the transmission within the expected time.

However, if a VM with dynamic guarantee does not send traffic at the beginning of transmission, the time to finish transmission t' decreases and the traffic size s' remains the same. For this case, the expected guarantee $b' = s'/t'$ will increase and even exceed the initial guarantee. Hence, we need to maintain the dynamic guarantee under the initial guarantee S/T, and only provide fairness guarantee after T.

4.3 Pricing Model

Usage-based charging. With bandwidth over commitment, the throughput of VMs may not achieve the guaranteed bandwidth (i.e., guarantee failure). To address this, SoftBW charges bandwidth according to the actual bandwidth usage. Suppose the unit price of strict bandwidth guarantee B_i is P_s. The VM with B_i bandwidth guarantee is charged $P_s \cdot B_i \cdot F_t$ at billing period t, where F_t is the fulfillment measured at period t. Traffic that exceeds B_i will be charged the same as the pricing of fairness guarantee (P_f), since it only gets a fair sharing. For example, in a billing cycle, if the throughput of a VM with 100 Mbps strict guarantee is 150 Mbps, the price will be $100P_s + 50P_f$. For dynamic guarantee, the unit price of bandwidth guarantee P_d relies on the average bandwidth guarantee $B_j = S/T$. The cost at period t is $P_d \cdot B_j \cdot F_t$. As dynamic guarantee may reduce the

Guarantee	Performance	Price
Strict	Bandwidth B	$r_t \cdot (1+B/C)P_0$
Dynamic	Data size S, time T	$r_t \cdot (1+S/TC)\beta P_0$
Fairness	VM-level fairness	$r_t \cdot r_t/C \cdot \beta P_0$
Best effort	No bandwidth guarantee	Free

Table 2: The price for bandwidth guarantee at a billing cycle: r_t is the measured rate of a VM.

failure rate in bandwidth guarantee under low network utilization situation, we set $P_d = \beta P_s$ ($\beta < 1$) to encourage tenants to use dynamic guarantee for massive delay-tolerant data transmission.

Performance-price consistency. However, for usage-based pricing, the tenants can declare higher bandwidth than their requirements to achieve higher performance under the same price, since the transmission time of the same size of data is proportionally reduced. For example, when transmitting 1 Gb data, using 100 Mbps bandwidth will cost 10 seconds, while using 200 Mbps bandwidth only costs 5 seconds. Both situations cost $1000P$, where P denotes the price of using 1 Mbps for 1 seconds. Hence, to keep performance-price consistency, the unit price of higher bandwidth guarantee should also be higher.

We use a non-decreasing pricing function for bandwidth guarantee, where $P_s = P_0(1+B/C)$, $P_d = (1+S/TC)\beta P_0$ (C represents the physical bandwidth, and P_0 is a constant price). Fairness guarantee has the lowest unit price $P_f = r_t/C \cdot \beta P_0$, which is always less than P_d. This way, tenants using strict guarantee will buy the lowest possible bandwidth according their requirements, and avoid unnecessary data transmission. For tenants using dynamic guarantee, the bandwidth guarantee becomes zero when finishing transmission of the declared data S. Thus, under-declaring the data size will not benefit the performance. In fact, they will transmit data as fast as possible, because their transmission costs will be cheaper if exceeding the expected bandwidth B_j. This also benefits the provider: the idle bandwidth is utilized and the dynamic guarantee decreases, thus more bandwidth can be used for other guaranteed traffic.

5 Fulfillment-based Scheduling

SoftBW applies a packet level scheduling in the agent to share bandwidth under over commitment based on the fulfillment abstraction in §3. SoftBW agent leverages the generalized processor sharing model [13] to serve the queue in a weighted round robin manner [14]. Thus, each queue gets a share of the bandwidth, and the fairness among VMs can be obtained even when bandwidth is constrained. For bandwidth guarantee, instead of using periodical rate measurement, the agent measures the transmission time of each packet, and only estimates whether the bandwidth guarantee of the VM is satisfied.

Our scheduling, the *estimation of fulfillment* and the *scheduling of packets*. In a round robin scheduling, the key task is to decide whether the packets at the head of queues should be transmitted at each round. We set up a timer to record the time-to-send (tts) for the packet at the head of the queue, which indicates the expected time point to transmit this packet if we want to meet the bandwidth guarantee. Then the scheduler decides whether the packet should be transmitted by comparing tts against the current time. Since the fulfillment of a queue decreases when it is waiting to be scheduled, the goal of fulfillment estimation is to update the time-to-send for the queues after each transmission.

5.1 Estimation of Fulfillment

For a queue, each time a packet (p_n) is transmitted, we calculate the inter-departure time between this packet and the last transmitted packet (p_{n-1}), denoted as τ. Let p_{size} denote the size of the packet (p_n). If the bandwidth guarantee for this queue is B, then the fulfillment can be expressed as $F = \frac{p_{size}/\tau}{B}$. Thus, for a VM whose bandwidth guarantee is not satisfied ($F < 1$), we can derive the following equation

$$\Delta\tau = \tau - \frac{p_{size}}{B} > 0, \tag{4}$$

where $\Delta\tau$ is the difference between the inter-departure time and the expected time of transmitting a packet with B. Since this difference in transmitting time means that the VM's rate is either larger ($\Delta\tau < 0$) or less ($\Delta\tau > 0$) than the guaranteed bandwidth, the inter-departure time should be accumulated in every update, so as to reduce the rate that is above the guaranteed bandwidth, as well as increasing the rate that is under the guaranteed bandwidth. Thus, $\overline{\Delta\tau} \leftarrow \overline{\Delta\tau} + \Delta\tau$. We maintain $\overline{\Delta\tau}$ in the interval $[-\tau_{max}, \tau_{max}]$ so that $\overline{\Delta\tau}$ will not infinitely decrease when bandwidth exceeds the guarantee, nor increase when bandwidth can not satisfy the guarantee.

Each time when $\overline{\Delta\tau}$ of a queue is re-calculated, we update the tts for this queue:

- If $\overline{\Delta\tau} \geq 0$, the bandwidth guarantee of the VM is not satisfied. Thus, we set tts to 0, to allow the scheduler to dequeue a packet from the queue.
- If $\overline{\Delta\tau} < 0$, the rate of the VM exceeds B. Then, the variable tts of the VMs is set to p_{size}/B ahead of current time:

  $$tts = t_{current} + \frac{p_{size}}{B}, \tag{5}$$

  which notifies the scheduler if a packet is transmitted before this time tts, the VM will exceeds the bandwidth guarantee (B).

- If $\overline{\Delta\tau}$ of a queue changes from positive to negative, it implies that the sending rate of the corresponding VM just exceeds its bandwidth guarantee, then we set

$$tts = t_{current} + \frac{p_{size}}{B} - \overline{\Delta\tau}. \quad (6)$$

  $\overline{\Delta\tau}$ is a compensation for the rate, since the VM's rate in previous round is less than the bandwidth guarantee.

5.2 Scheduling of Packets

Before scheduling a packet, the scheduler first compares the current time $t_{current}$ against the tts of a queue. There are three conditions to consider:

- If $tts = 0$, then the rate of the VM is below the bandwidth guarantee and the scheduler dequeues the packet at the head of the queue.
- If $0 < tts \leq t_{current}$, the scheduler has just missed the expected transmission time. If the packet is transmitted at the current time, the VM's rate will not exceed the guaranteed bandwidth. Hence, the scheduler can transmit a packet from this queue.
- If $tts > t_{current}$, then the VM will exceed the bandwidth guarantee after we send a packet. For this case, the scheduler will check the status of the physical bandwidth and only sends a packet if there is any residual bandwidth on this server.

Work-conserving. Similar to queues of VMs, the status of the physical bandwidth is maintained by calculating the difference ($\Delta\tau_c$) between the inter-departure time and the expected time of transmitting a packet with the maximal physical rate, after transmitting a packet from any queue. When $\Delta\tau_c > 0$, which indicates the physical bandwidth is not fully utilized, scheduler can still transmit packets from queues that have exceeded the bandwidth guarantee. This way, the residual bandwidth of the host server can be allocated if there is unsatisfied traffic demand, thus the bandwidth sharing is work-conserving.

Performance guarantee. The round robin scheduler can preferentially transmit the packets from VMs whose bandwidth guarantee is not satisfied. As a result, the rates of these VMs can quickly increase even when the physical bandwidth are taken by other traffic. For example, when a VM with a bandwidth guarantee starts to send traffic on a fully utilized access link, the newly arrived packets can be transmitted at each round as the VM's fulfillment is less than 1 and tts is 0. Note that this also benefits the performance of the short flows, since they can transmit packets without waiting for other VMs to decrease their rates. For fairness guarantee, we need to set a small bandwidth guarantee for the queues, so that their traffic will not be blocked when the bandwidth is fully utilized by guaranteed traffic.

6 Evaluation

In this section, we evaluate the performance of SoftBW from the following aspects:

- *Performance guarantee*: We validate that SoftBW can achieve stable bandwidth guarantee and at the same time, maintain fairness among VMs even when bandwidth is over committed.
- *Fast allocation*: We validate that SoftBW has the property of small convergence time (~ 10 ms) in the presence of highly bursty traffic, and $2.8\times \sim 4.5\times$ improvement on the completion time for short flows as compared with the rate-limit approach.
- *Overhead*: We analyze the overhead of SoftBW and find that SoftBW adds less than 1.9 μs transmission delay to the TCP's RTT, and maintains less than 5.1% CPU overhead under 10 Gbps transmission.
- *Over commitment*: We examine the impact of bandwidth over commitment on resources sharing and show that the provider can possibly increase average $3.9\times$ network utilization while maintaining the average failure rate within 5% in our simulation.

6.1 Evaluation Setup

We first perform testbed experiments to evaluate SoftBW's performance on bandwidth guarantee (§6.2) and the overhead of scheduling (§6.3). We then use simulation to study the impact of over commitment on large scale (§6.4).

Testbed. The testbed consists of 14 servers. Each server has an Intel Xeon E5-2670 2.6 Ghz CPU (8 physical cores with hyper-threading) and an Intel 82580 Gigabit NIC connected to a 1 GbE switch port. The servers run Linux 2.6.32 kernel, among which one acts as the controller with OpenDaylight and the others host servers with KVM and Open vSwitch (OVS). Each VM has a `virtio` NIC with `vhost-net` enabled, connecting to a `tap` device attached to an OVS bridge. We compare SoftBW with an existing rate-limit based bandwidth allocation [5], represented as ES (ElasticSwitch).

Simulator. We simulate a 2,000-server datacenter with full bisection bandwidth. Each server connects to the switch with a 1 Gbps link. The strict guarantee and the initial expected guarantee for dynamic guarantee are both 250 Mbps. Thus without over commitment, each server can deploy 4 VMs. As we focus on network-intensive applications, the simulator only considers network bandwidth, and allocates bandwidth at 1s interval.

Workload. In the simulation, we use two different traffic loads: 1) For strict guarantee, the traffic demand of a VM follows a exponential distribution around a mean of 250ρ Mbps on each time slot. 2) For dynamic guarantee, the data size in a VM follows a exponential dis-

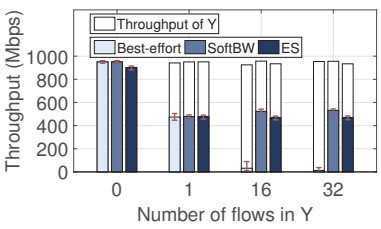

Figure 5: Throughput of X with increasing number of flows in Y. Both VMs have 450 Mbps guarantees.

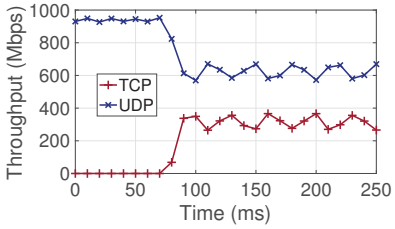

Figure 6: The convergence process of X when Y sends a bursty UDP flow on the same congested link.

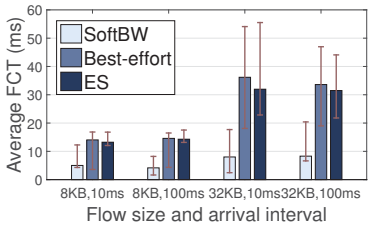

Figure 7: Guarantee for short flows: SoftBW improves the average flow completion time (FCT).

tribution with a mean of $250\rho \cdot T_{max}$ Mbits, and the start time follows a Poisson process with N_d/T_{max} arrival rate. (T_{max}: simulation time, N_d: #VMs with dynamic guarantee.) The source and destination of traffic on each VM are chosen uniformly at random. Thus the expected network utilization without over commitment will be ρ.

Parameters. The evaluation chooses 1 second as the time interval of updating the dynamic bandwidth guarantee. Note that it is a suitable setting when compared with VM lifetime. As TCP flows can achieve at most 940 Mbps in our testbed, the maximum guarantee without over commitment is set as 90% of 1 Gbps physical bandwidth on each server.

6.2 Bandwidth Guarantee and Efficiency

Bandwidth guarantee. We co-locate two sender VMs on one server: VM X connects to a receiver with one TCP flow, and VM Y connects to N receivers, each with one flow. Both VM X and Y have 450 Mbps strict bandwidth guarantee. We vary N, the number of receivers of Y, and show the throughput of VM X in Figure 5. The left bar in each group represents the throughput without guarantee, and X suffers unfairness when Y has more flows. With SoftBW's scheduling, X is guaranteed with a rate of $\sim$450 Mbps when Y has multiple flows, and utilizes the entire link when Y has no traffic. This verifies SoftBW's work-conserving property and the ability of enforcing bandwidth guarantee under aggressive bandwidth competitions.

Convergence process. We show SoftBW's adaption to sudden traffic changes by quantifying the convergence process of long flows. When VM X (with 600 Mbps guarantee) sends traffic with a long-lived flow to a remote receiver and becomes stable, VM Y (with 300 Mbps guarantee) starts to generate bursty UDP traffic to another receiver with 800 Mbps sending rate. Figure 6 shows the throughput of X and Y measured at the receiving end. When the traffic in Y arrives, it consumes the bandwidth in around 10 ms, which demonstrates SoftBW's fast convergence on re-allocating the utilized bandwidth. Due to TCP's rate control, we observe fluctuations at around 10 ms timescale, however, the average throughput measured by every 100 ms is stable , which is sufficient for usage-based charging.

Short flows. Since short flows' durations are too short to fully utilize the guaranteed bandwidth, we quantify SoftBW's guarantee for short flows by examining the completion time of these flows when competing with existing long flows. Figure 7 illustrates the scenario where VM Y (450 Mbps bandwidth guarantee) is continuously sending traffic, and VM X (450 Mbps bandwidth guarantee) generates short flows on the same congested link. The short flows are of 8 KB/32 KB in size, and the flow inter-arrival times are 10 ms/100 ms. Without scheduling, the increase of flow rate relies on creating a congestion on the link which notifies the existing flows to adjust their rates (best-effort), or rate-limits those flows to decrease their sending rate (ES). These adjustment may take a long time for a short flow to acquire the necessary bandwidth resource. When enabling SoftBW, the packets from short flows can quickly obtain the bandwidth, since X's fulfillment is less than Y and so X's packets will be scheduled without delay at each round. As a result, the flows in X are guaranteed to have a higher average rate, thus the completion time is $2.8\times\sim4.5\times$ shorter than that of rate-limiting or best effort packet scheduling.

Over commitment. We set up an over committed scenario where three VMs each with 450 Mbps strict guarantee are sharing a 1 Gbps link. In the worst case, when all VMs are send traffic continuously, the total traffic demand exceeds 1 Gbps and their bandwidth can not be guaranteed. Figure 8 shows the rates of VMs when the ratio of the number of flows in each VM is 1 : 1 : 2. Since SoftBW uses per-VM queue, each VM obtains about 300 Mbps bandwidth, and the fairness among VMs is guaranteed irrespective of the flows in VMs. However, since the rate-limit based guarantee policy relies on limiting the rate beyond the minimum guarantee, it can not maintain fairness under this condition where the rate of each VM is less than the bandwidth guarantee, thus VM with more flows receives more bandwidth.

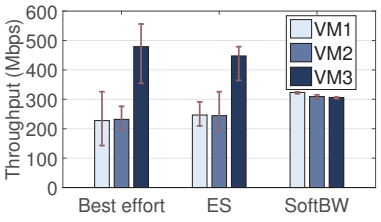

Figure 8: SoftBW provides fairness among VMs' throughput under bandwidth over commitment, where each VM has 450 Mbps guarantee.

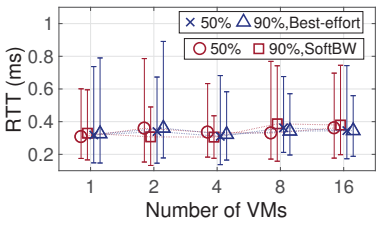

Figure 9: SoftBW does not add latency to RTT under 50% and 90% UDP traffic loads as compared to best-effort manner.

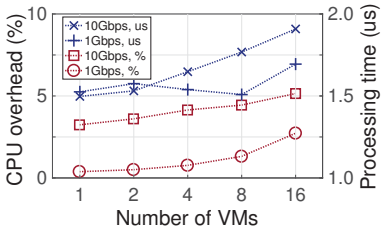

Figure 10: SoftBW's CPU overhead and processing time of each packet with different numbers of VMs on each server.

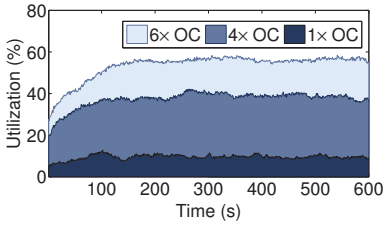

Figure 11: The network wide utilization in 600 s simulation under 1×, 4× and 6× OC.

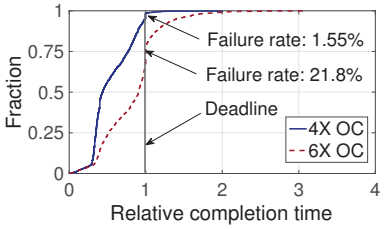

Figure 12: The relative completion time of data transmission using dynamic guarantee under 4× and 6× OC.

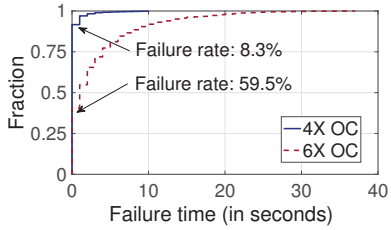

Figure 13: The failure time (in seconds) of VMs using strict guarantee under 4× and 6× OC.

6.3 Overhead Analysis

We evaluate SoftBW's latency overhead in comparison to the best effort manner without any scheduling. As we focus on network intensive applications, the packets from traffic generator have a size of MTU (1500 Bytes), which can achieve 10 Gbps throughput with only one CPU core. Hence, network is the only bottleneck in the experiments. For TCP delay, we leave a 10% gap between the maximum workloads and the physical bandwidth to reduce the impact of link congestion on RTT. The number of VMs is capped by the vCPUs on each server, i.e., 16.

Latency. Figure 9 shows the RTTs between two VMs on different servers, with 50% and 90% UDP traffic loads. As the traffic load and the number of VMs increase, we see no obvious increase of latency in RTT (with traffic less than 1 Gbps). The fluctuation of RTT is also small enough to maintain the stability of TCP flows. We attribute this to the high efficiency of OVS's software tunneling supported by today's powerful processors. At the same time, SoftBW's scheduling, which only adds at most five steps of floating point computation for each packet, will not be a bottleneck for packet transmission as compared to the VM-to-VM latency ($\sim 350\,\mu$s in the same availability zone of EC2).

CPU overhead. Figure 10 shows the CPU overhead and processing time of each packet with SoftBW. We conduct traffic by sending data from VMs to their host server, where the total throughput is rate limited under 10 Gbps by TC. As expected, the CPU overhead increases as we increase the total throughput and the number of VMs on each server. The maximum throughput reaches ~ 9 Gbps and the overhead with 16 VMs is only 5.1%. Considering that the access bandwidth in current data center are often 10 Gbps, this overhead is acceptable for providers. The processing time of each packet in the scheduling is about 1.5 μs. Due to the contention of CPU, the processing time increases as the number of VMs increases. This latency in scheduling is much less than TCP RTT, hence will not degrade the performance of underlying TCP flows.

6.4 Bandwidth Over Commitment

When bandwidth is over committed, the guarantee failure happens under two conditions: First, for a VM with strict guarantee, traffic demand is not satisfied when it is less than the bandwidth guarantee. Second, for dynamic guarantee, the traffic is not finished before the deadline. The simulator runs 600 s with $\rho = 15\%$, where we record the rate and transmission time of each VM.

Network utilization. Figure 11 shows the network wide utilization under different over commitment. The average utilization without over commitment is about 9.5%. When bandwidth is 4× OC, there is a remarkable improvement on the average utilization by 3.9×, from 9.5% to 37.4%. Using more aggressive OC (6×) can further increase the network utilization, however, as we will

show, also brings significant failures to the bandwidth or deadline guarantee.

Guarantee failures. Figure 12 shows the relative completion times (in percentile of guaranteed completion time) of data transmission with dynamic guarantee. Figure 13 shows the failure time of VMs with strict guarantee. With 4× OC, 98.4% VMs finish transmission before their deadlines under dynamic guarantee. Only 1.6% VMs fails, and the worst performance is 2.0× longer than the deadline. For VMs with strict guarantee, only 8.4% of them experience guarantee failure. The total failure duration of each VM is also very low, among which the longest one is 10 s during 600 s simulation. As a result, when the average network utilization is around 10%, it is feasible for providers to use 4× OC for bandwidth guarantee at a large scale, because there is little chance to encounter insufficient bandwidth, and the worst performance with guarantee failure is also acceptable. Thus the provider can consider to compensate the tenants for the guarantee failures at a low cost. When the over commitment increases to 6×, we can observe obvious guarantee failure for both strict guarantee and dynamic guarantee, whose failure rate are 59.5% and 21.8%, respectively. Hence, it is not suitable for this traffic load. The simulation validates the feasibility of bandwidth over commitment in multi-tenant clouds, and provides a solution to estimate the proper over commitment ratio by tracking the network utilization.

7 Related Work

Bandwidth allocation. The first piece of work focuses on bandwidth reservation in datacenters [10, 15, 16, 4]. By proposing performance models and VM allocation mechanisms, they allocate the VMs to servers which can meet the performance requirements. Another policy is to slice the physical bandwidth according to the traffic demand of VMs using rate-control [7, 17, 18]. The two solutions are complementary to each other, since the bandwidth of VMs can be re-shaped after allocation.

Faircloud [7] presents the bandwidth requirements of bandwidth allocation problem and develops traffic slicing strategies for proportional sharing. NetShare [19] achieves proportional bandwidth sharing among different tenants by using weighted fair queues in switches. SoftBW uses virtual switches in the hypervisor, hence, the traffic with bandwidth guarantee will not be limited by the number of hardware queues in the switches.

Seawall [20] leverages end-based rate limit to achieve VM-level weighted max-min fairness. This policy can be extended for bandwidth guarantee with a lower-bounded rate-limit for VMs [5, 21]. ElasticSwitch [5] partitions the bandwidth guarantees of VMs to VM-pairs, and achieves minimum guarantee using a TCP-Cubic based rate control for VM-to-VM traffic. EyeQ [11] measures rate every 200 μs at the receivers, and uses this information to enforces the rate of senders to achieve minimum guarantee. eBA [22, 23] uses the feedback of link utilization from switches to control the rate of senders. The rate-limiting based solution assumes that the total bandwidth guarantee is less than the physical bandwidth, and is not suitable for the cloud providers to over commit their bandwidth.

Bandwidth pricing. Usage-based pricing model is widely used in current IaaS clouds [24]. Recent proposals have studied the economical impact of cloud resource pricing on system design and providers' revenue [25, 26, 27]. We target at providing price-performance consistency, and improving providers' revenue is included in our future work. [28] answers the question of how users should bid for cloud spot instances, which aims at saving costs for users. [29] discusses dynamic pricing for inter-datacenter traffic, rather than intra-datacenter network bandwidth. The solutions are not suitable for our situation, since their pricing for bandwidth does not involve over commitment on bandwidth guarantee.

Scheduling/congestion control. SoftBW's scheduling framework is based on previous round robin scheduling such as, CBQ [30], Fair queueing [13], and adds mechanisms to enforce fairness of fulfillments. There are also a number of works , which use scheduling (e.g., [31, 32, 33]) or new congestion control (e.g., [34, 35, 36])) to improve the performance of flows and reduce the latency in datacenters. Virtual congestion control [37, 38] can help to deploy new congestion control in the hypervisors without changing VM network stack. Our work is complementary to these works, which focus on the performance at flow level, while we provide bandwidth guarantee and pricing for VMs.

8 Conclusion

In this paper, we presented SoftBW, a solution that enables pricing bandwidth for VMs in IaaS datacenters, by providing efficient bandwidth/fairness guarantee with bandwidth over commitment. SoftBW's over commitment on bandwidth is rational, since the failure rate of bandwidth guarantee can be controlled to a low level by conservatively choosing the over commitment ratio based on the network utilization. SoftBW's design is easy to be implemented, since it uses software virtual switches at each server to schedule VMs' traffic and can be centrally controlled. SoftBW applies a usage-based charging, which is deployable for current charging model in multi-tenant clouds, thus giving a feasible solution for providers to realize quantified bandwidth performance and pricing for cloud VM instances.

References

[1] ROY, A., ZENG, H., BAGGA, J., PORTER, G., AND SNOEREN, A. C. Inside the social network's (datacenter) network. In *ACM SIGCOMM Computer Communication Review* (2015), vol. 45, ACM, pp. 123–137.

[2] BENSON, T., AKELLA, A., AND MALTZ, D. A. Network traffic characteristics of data centers in the wild. In *ACM IMC* (2010).

[3] KANDULA, S., SENGUPTA, S., GREENBERG, A., PATEL, P., AND CHAIKEN, R. The nature of data center traffic: measurements & analysis. In *ACM IMC* (2009).

[4] LEE, J., TURNER, Y., LEE, M., POPA, L., BANERJEE, S., KANG, J.-M., AND SHARMA, P. Application-driven bandwidth guarantees in datacenters. In *ACM SIGCOMM* (2014).

[5] POPA, L., YALAGANDULA, P., BANERJEE, S., MOGUL, J. C., TURNER, Y., AND SANTOS, J. R. Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing. In *ACM SIGCOMM* (2013).

[6] GUO, J., LIU, F., ZENG, D., LUI, J. C., AND JIN, H. A cooperative game based allocation for sharing data center networks. In *IEEE INFOCOM* (2013).

[7] POPA, L., KUMAR, G., CHOWDHURY, M., KRISHNAMURTHY, A., RATNASAMY, S., AND STOICA, I. Faircloud: Sharing the network in cloud computing. In *ACM SIGCOMM* (2012).

[8] GREENBERG, A., HAMILTON, J., JAIN, N., KANDULA, S., KIM, C., LAHIRI, P., MALTZ, D., PATEL, P., AND SENGUPTA, S. Vl2: a scalable and flexible data center network. In *ACM SIGCOMM* (2009).

[9] AL-FARES, M., LOUKISSAS, A., AND VAHDAT, A. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication* (New York, NY, USA, 2008), SIGCOMM '08, ACM, pp. 63–74.

[10] BALLANI, H., COSTA, P., KARAGIANNIS, T., AND ROWSTRON, A. Towards predictable datacenter networks. In *ACM SIGCOMM* (2011).

[11] JEYAKUMAR, V., ALIZADEH, M., MAZIÈRES, D., PRABHAKAR, B., KIM, C., AND GREENBERG, A. Eyeq: practical network performance isolation at the edge. In *USENIX NSDI* (2013).

[12] Amazon ec2 service level agreement. http://aws.amazon.com/ec2/sla/.

[13] SHREEDHAR, M., AND VARGHESE, G. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM Computer Communication Review* (1995).

[14] SHREEDHAR, M., AND VARGHESE, G. Efficient fair queueing using deficit round robin. *SIGCOMM Comput. Commun. Rev. 25*, 4 (Oct. 1995), 231–242.

[15] GUO, C., LU, G., WANG, H., YANG, S., KONG, C., SUN, P., WU, W., AND ZHANG, Y. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *ACM CoNEXT* (2010).

[16] XIE, D., DING, N., HU, Y., AND KOMPELLA, R. The only constant is change: incorporating time-varying network reservations in data centers. In *ACM SIGCOMM* (2012).

[17] GUO, J., LIU, F., LUI, J. C. S., AND JIN, H. Fair network bandwidth allocation in iaas datacenters via a cooperative game approach. *IEEE/ACM Trans. Networking 24*, 2 (Apr. 2016), 873–886.

[18] GUO, J., LIU, F., TANG, H., LIAN, Y., JIN, H., AND LUI, J. C. Falloc: Fair network bandwidth allocation in iaas datacenters via a bargaining game approach. In *IEEE ICNP* (2013).

[19] LAM, T., AND VARGHESE, G. Netshare: Virtualizing bandwidth within the cloud. Tech. rep., UCSD, 2009.

[20] SHIEH, A., KANDULA, S., GREENBERG, A., KIM, C., AND SAHA, B. Sharing the data center network. In *USENIX NSDI* (2011).

[21] RODRIGUES, H., SANTOS, J., TURNER, Y., SOARES, P., AND GUEDES, D. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *USENIX Workshop on I/O Virtualization* (2011).

[22] GUO, J., LIU, F., HUANG, X., LUI, J. C., HU, M., GAO, Q., AND JIN, H. On efficient bandwidth allocation for traffic variability in datacenters. In *IEEE INFOCOM* (2014).

[23] LIU, F., GUO, J., HUANG, X., AND LUI, J. C. S. eba: Efficient bandwidth guarantee under traffic variability in datacenters. *IEEE/ACM Trans. Netw. 25*, 1 (Feb. 2017), 506–519.

[24] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., AND ZAHARIA, M. A view of cloud computing. *Commun. ACM 53*, 4 (Apr. 2010), 50–58.

[25] NIU, D., FENG, C., AND LI, B. Pricing cloud bandwidth reservations under demand uncertainty. In *ACM SIGMETRICS Performance Evaluation Review* (2012).

[26] WANG, H., JING, Q., CHEN, R., HE, B., QIAN, Z., AND ZHOU, L. Distributed systems meet economics: pricing in the cloud. In *USENIX HotCloud* (2010).

[27] XU, H., AND LI, B. A study of pricing for cloud resources. *ACM SIGMETRICS Performance Evaluation Review* (2013).

[28] ZHENG, L., JOE-WONG, C., TAN, C. W., CHIANG, M., AND WANG, X. How to bid the cloud. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (New York, NY, USA, 2015), SIGCOMM '15, ACM, pp. 71–84.

[29] JALAPARTI, V., BLIZNETS, I., KANDULA, S., LUCIER, B., AND MENACHE, I. Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference* (New York, NY, USA, 2016), SIGCOMM '16, ACM, pp. 73–86.

[30] FLOYD, S., AND JACOBSON, V. Link-sharing and resource management models for packet networks. *IEEE/ACM TON* (1995).

[31] AL-FARES, M., RADHAKRISHNAN, S., RAGHAVAN, B., HUANG, N., AND VAHDAT, A. Hedera: Dynamic flow scheduling for data center networks. In *NSDI* (2010).

[32] ALIZADEH, M., YANG, S., SHARIF, M., KATTI, S., MCKEOWN, N., PRABHAKAR, B., AND SHENKER, S. pfabric: Minimal near-optimal datacenter transport. *ACM SIGCOMM* (2013).

[33] CHOWDHURY, M., ZHONG, Y., AND STOICA, I. Efficient coflow scheduling with varys. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (New York, NY, USA, 2014), SIGCOMM '14, ACM, pp. 443–454.

[34] ALIZADEH, M., GREENBERG, A., MALTZ, D. A., PADHYE, J., PATEL, P., PRABHAKAR, B., SENGUPTA, S., AND SRIDHARAN, M. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference* (New York, NY, USA, 2010), SIGCOMM '10, ACM, pp. 63–74.

[35] JANG, K., SHERRY, J., BALLANI, H., AND MONCASTER, T. Silo: Predictable message latency in the cloud. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (New York, NY, USA, 2015), SIGCOMM '15, ACM, pp. 435–448.

[36] NAGARAJ, K., BHARADIA, D., MAO, H., CHINCHALI, S., ALIZADEH, M., AND KATTI, S. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference* (New York, NY, USA, 2016), SIGCOMM '16, ACM, pp. 188–201.

[37] CRONKITE-RATCLIFF, B., BERGMAN, A., VARGAFTIK, S., RAVI, M., MCKEOWN, N., ABRAHAM, I., AND KESLASSY, I. Virtualized congestion control. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference* (New York, NY, USA, 2016), SIGCOMM '16, ACM, pp. 230–243.

[38] HE, K., ROZNER, E., AGARWAL, K., GU, Y. J., FELTER, W., CARTER, J., AND AKELLA, A. Ac/dc tcp: Virtual congestion control enforcement for datacenter networks. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference* (New York, NY, USA, 2016), SIGCOMM '16, ACM, pp. 244–257.