Sun *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2021) 10:55
https://doi.org/10.1186/s13677-021-00271-5

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

A price-aware congestion control protocol for cloud services

Xiaocui Sun[1,2*], Zhijun Wang[3], Yunxiang Wu[4], Hao Che[3] and Hong Jiang[3]

Abstract

In current infrastructure-as-a service (IaaS) cloud services, customers are charged for the usage of computing/storage resources only, but not the network resource. The difficulty lies in the fact that it is nontrivial to allocate network resource to individual customers effectively, especially for short-lived flows, in terms of both performance and cost, due to highly dynamic environments by flows generated by all customers. To tackle this challenge, in this paper, we propose an end-to-end Price-Aware Congestion Control Protocol (PACCP) for cloud services. PACCP is a network utility maximization (NUM) based optimal congestion control protocol. It supports three different classes of services (CoSes), i.e., best effort service (BE), differentiated service (DS), and minimum rate guaranteed (MRG) service. In PACCP, the desired CoS or rate allocation for a given flow is enabled by properly setting a pair of control parameters, i.e., a minimum guaranteed rate and a utility weight, which in turn, determines the price paid by the user of the flow. Two pricing models, i.e., a coarse-grained VM-Based Pricing model (VBP) and a fine-grained Flow-Based Pricing model (FBP), are proposed. The optimality of PACCP is verified by both large scale simulation and small testbed implementation. The price-performance consistency of PACCP are evaluated using real datacenter workloads. The results demonstrate that PACCP provides minimum rate guarantee, high bandwidth utilization and fair rate allocation, commensurate with the pricing models.

Keywords: Pricing model, Cloud computing, Congestion control, Network utility maximization

Introduction

An infrastructure-as-a-service (IaaS) cloud, such as Amazon EC2 and Alibaba cloud, provides scalable, pay-as-you-go computing resources to its customers. However, to date, the customers using such cloud services are charged based on the usage of the computing related resources only, e.g., various instances of virtual machines (VM) with different reserved CPU and memory resources. This, however, is inadequate, as paying for a given VM instance provides no assurance of flow performance for a flow emitted from that instance. The measurements of bandwidth usage for VMs in public cloud platforms show that the network performance is independent on the price of VM [1], the VMs with the same price may have huge different network performances and a cheaper VM can have better network performance than an expensive VM. The root cause of the status quo is that the network bandwidth is shared in a highly dynamic environment by flows emitted from all VM instances and hence, it is difficult to provide quantifiable flow rate allocation in a cost-effective fashion so that an effective pricing structure can be built around it. A direct consequence for not being able to do so is that a customer may experience poor performance, especially at high network utilization, incommensurate with the price the customer has paid for the use of the computing resources[1]. Hence the pricing model only based on computing resource usage may not be suitable for tenants who need guaranteed network performance. In fact, they would like to pay higher price to receive highly guaranteed network performance to attractive their customers and increase their business profits.

*Correspondence: xiaocuisun1002@hotmail.com
[1]Department of Computer Science, Guangdong Pharmaceutical University, Guangzhou, China
[2]Medical Information and Real World Engineering Technology Center, Guangdong Pharmaceutical University, Guangzhou, China
Full list of author information is available at the end of the article

To tackle the above challenges, network pricing solutions based on explicit bandwidth reservation have been proposed [2–14]. The price is usually dynamically generated either through an auction process [3, 5, 7, 8, 10–13] or a time-varying price table [2, 4, 6, 9, 14], adjusted based on the current and/or historical statistics. However, these pricing solutions are only effective for long-lived flows, such as video on demand [15], not for the popular user-facing datacenter applications [16–20] which usually have small flow sizes and short durations.

User-facing datacenter applications, such as Web searching [17] and social networking [19], are usually associated with a stringent tail-latency service level objective (SLO) [21]. Moreover, a job for such an application generally involves one or multiple stages of parallel task processing by (many) instances, which generate bursts of (massive) numbers of flows emitted from those instances. Such flows are usually short-lived with sizes of less than 1 Mbytes [22, 23] and with tight flow completion time budget, e.g., a few milliseconds, to meet a prescribed tail-latency SLO.

The pricing solutions mentioned earlier are based on centralized bandwidth reservation, which is either pre-configured or flow-driven, none of which however, can deal with the above workload effectively. On one hand, pre-configured bandwidth reservation that allocates bandwidths for the prospective flows in advance is not scalable and cannot handle bursts of massive numbers of short-lived flows. Moreover, without knowing the flow start time and flow size, this approach may lead to either over or under resource provisioning, causing violation of SLOs or low resource utilization, respectively. On the other hand, flow-driven bandwidth reservation that reserves bandwidth upon a flow arrival is generally too slow due to centralized control to meet the tight flow response time budget of such flows and incurs excessive processing and communication overheads. Moreover, these solutions need significant core network switches modification/upgrading incurring high costs. Although a price-aware distributed scheduling solution, known as SoftBW [1], is proposed to allow scalable bandwidth reservation with flow rate guarantee, it only works at very low network loads (less than 30%), as it reserves bandwidths for individual VM instances at each host port only assuming that the datacenter network is congestion free. Any retransmission can result in flow rate allocation inappropriate for the paid prices.

A desirable solution for sharing network bandwidth in cloud services should meet the widely accepted requirements [2]. At first it should be easily implemented. The solution should not require significant hardware/software modification in current datacenters. Secondly, it should be scalable and maintain high network bandwidth utilization. The third one is that it should provide different prices to meet different tenant demands. For example, some tenants need to meet their strict flow deadlines, and hence they may ask for minimum rate guaranteed services; while the others may expect short flow completion times without strict deadline requirement, and hence they may ask for differential services. The fourth and the last one is that the rate allocation should be proportional to the price paid by a tenant.

To overcome the above shortcomings of the existing network pricing solutions and meet the widely accepted requirements of a desirable solution, in this paper, we propose a Price-Aware Congestion Control Protocol (PACCP) for IaaS cloud services. PACCP is a network utility maximization (NUM) based optimal congestion control protocol. It supports three different class of services (CoSes), i.e., best effort service (BE), differentiated service (DS) and minimum rate guaranteed (MRG) service. The three types of services are enabled by properly setting the values of a pair of parameters, i.e., a minimum guaranteed rate and a utility weight, which are, in turn, determined by the flow price paid for the services.

In this paper, we propose two pricing models, i.e., a coarse-grained VM-Based Pricing model (VBP) [1] and a fine-grained Flow-Based Pricing model (FBP). A tenant pays a price to buy a desired service, which is then mapped to given values of the pair of parameters in PACCP. PACCP possesses the following salient features,

- It is an optimal solution in terms of network utility maximization (NUM); It uses the TCP utility function and hence it is a TCP friendly protocol.
- It meets the widely accepted requirements for datacenter price-based rate allocation solutions mentioned above [2], i.e., easy implementation; providing minimum rate guarantee; achieving high network utilization; and allocating flow rates in proportion to the paid prices;
- To the best of our knowledge, it is the first solution that seamlessly integrates pricing models with end-to-end congestion control protocols. Hence, it is high scalable and can deal with bursts of unlimited numbers of short-lived flows. It allows flows to fully utilize all available bandwidths and thus improving the bandwidth utilization. Moreover, it allows adjustment of pricing at runtime, adapting to resource demand changes and/or network load changes;
- It only requires software upgrade in end hosts and does not need any change in core network switches and hence, is readily deployable in today's datacenters.

The optimality and the price-performance consistency of PACCP are verified and evaluated by large scale simu-

[1] As container/docker is widely used now, all techniques used for VM can be applied to container/docker, to be simple, we just use VM in this paper.

lations as well as a small testbed implementation. The results demonstrate that PACCP can indeed provide soft minimum rate guarantee, high network utilization and rate allocation proportional to the prices paid, hence, meeting all the requirements for datacenter network pricing solutions. A preliminary version of this paper appeared in [24].

Network utility maximization based rate allocation

Assume that a network has n active flows and $U_i(x_i)$ is the user utility function of flow rate x_i for flow i (i=1,2,...,n). Then NUM is defined as the following,

$$V = max\left\{\sum_{i=1}^{n} U_i(x_i)\right\}, \tag{1}$$

subject to link bandwidth constraint,

$$\sum_{i=1}^{n} x_i \le B_l \quad \forall l \tag{2}$$

here B_l is the link bandwidth for link l and the flow rate constraint,

$$x_i \le \theta_i \quad \forall i \tag{3}$$

where θ_i is the required minimum flow rate for flow i, it is 0 if the flow has no minimum rate requirement.

The goal of NUM is to find distributed flow rate control laws that lead to flow rate allocation, $\mathbf{x}$= $\{x_1, x_2, ..., x_n\}$, where the global design objective V is attained, or collective user satisfaction of the services is maximized, as user utilities are meant to characterize to what degree users are satisfied with the services they receive. Different applications may have different user utility functions [25]. Clearly, the relative user utilities of the flows determine the rate allocation $\mathbf{x}$, provided that the flow rate constraints are satisfied. In other words, in NUM, the fairness criterion is uniquely determined by the relative user utilities of the flows. While minimum flow rate requirements can be easily enforced as flow rate constraints in NUM, it is nontrivial to enable flexible and quantifiable fairness criteria among flows with different user utility functions as the traditional NUM usually can only work on a single user utility function. Our recently developed NUM solution, called HOLENT NUM [26], can deal with different user utility functions by setting different utility weights on a base user utility function. We apply the similar idea by using a flow price to set up a flow weight. In what follows, we propose our solution based on price aware weighted user utilities.

We consider the following weighted utility function, i.e., $U_i(x_i) = \omega_i U_0(x_i)$, where U_0 is a base utility function shared by all the flows and ω_i is the weight of flow i ($i = 1, 2, ..., n$). To be backward compatible with and friendly to TCP flows, we use the TCP utility function (U_{TCP}), which is concave, as the base utility function. The utility function for TCP Reno is derived in [27, 28] and is given as follows. In the slow start phase (SSP),

$$U_{TCP}(x) = xlog\left(1 + \frac{\alpha}{\beta}\right), \tag{4}$$

and, in congestion avoidance phase (CAP),

$$U_{TCP}(x) = \left(\frac{\mu}{\beta} + x\right)[\,log(\mu+\beta x)-1] - x[\,log(\beta x)-1], \tag{5}$$

where αx and βx are the multiplicative increase and decrease rates, respectively. To match with SSP in TCP Reno where the flow rate is doubled/halved every round trip time (RTT), we have α=2β=1/RTT, by approximating the increase and decrease rates to be constant within a RTT interval. μ is the additive-increase rate (i.e., the rate of one packet per RTT) in CAP. With this TCP utility, now NUM can be rewritten as,

$$V = max\left\{\sum_{i=1}^{n} \omega_i U_{TCP}(x_i)\right\}, \tag{6}$$

subject to the link bandwidth and minimum flow rate constraints.

Now the idea is to enable flexible fair flow rate allocation through weight assignment. Specifically, consider flows with different weights sharing a bottleneck link l. Define a Lagrangian function

$$L(\mathbf{x}) = \sum_{i=1}^{n} \omega_i U_{TCP}(x_i) - \lambda\left(B_l - \sum_{i=1}^{n} x_i\right), \tag{7}$$

With the Lagrangian multiplier technique [29], the optimal solution is $\nabla_{\mathbf{x}} L(\mathbf{x}) = \mathbf{0}$. Consider two flows i and j with weight ω_i and ω_j, respectively, in the n active flows, we have

$$\frac{\partial L(\mathbf{x})}{\partial x_i} = \frac{\partial L(\mathbf{x})}{\partial x_j} = 0 \quad \forall i, j. \tag{8}$$

From Eq. (8), we can easy show that the rate allocation has to meet the following condition[2],

$$\frac{\omega_i}{\omega_j} = \frac{\partial U_{TCP}(x_j)/\partial x_j}{\partial U_{TCP}(x_i)/\partial x_i} = \frac{log(1+\mu/\beta x_j)}{log(1+\mu/\beta x_i)} \approx \frac{x_i}{x_j}, \quad \forall i, j, \tag{9}$$

for any pair of flows i and j bottlenecked at this link. Here we assume $\beta x \gg \mu$, i.e., the multiplicative decrease rate (i.e., half of the flow rate) is much larger than the additive increase rate (rate of 1 packet per RTT), and hence $log(1 + \mu/\beta x) \approx \mu/\beta x$. Eq. (9) clearly indicates that the allocated flow rate ratio is proportional to their utility

[2] Here we apply the CAP TCP utility, not the SSP TCP utility, because the TCP timeout is a rare event and TCP runs in the congestion avoidance phase most of the time.

weight ratio for any two flows sharing a bottleneck link. Hence, the relative flow rates allocated to different flows can be easily adapted to the fairness criterion underlaid by any given pricing model, through the proper setting of the corresponding relative weights.

NUM based optimal congestion control laws

A family of optimal congestion control laws to NUM with concave user utilities are derived by Su, et. al. [30], which underpins PACCP. Now we first introduce this family of optimal congestion control laws, which is then applied to the NUM problem in Eq. (6) to derive PACCP.

For simplicity, the subscript i for flow i is skipped hereafter. For any flow with concave utility function $U(x)$, the family of optimal congestion control laws that satisfies V are,

$$\dot{x} = z(x,t,cg)[f(x) - (1 - \bar{cg}r(x))] \tag{10}$$

with

$$f(x) = 1 - e^{-dU(x)/dx}, \tag{11}$$

where $z(x,t,cg)$ can be any piecewise continuous positive scaler function, resulting in an unlimited number of possible control laws in the family, it can be constructed to contain the path congestion information such as RTT and explicit congestion notification (ECN); cg is the path congestion indicator, taking value 1, if the path is congested, and 0, otherwise; $\bar{cg}$ is the logical negation of cg; $r(x)$ is a scalar parameter associated with the minimum rate requirement. Assume that a flow has a minimum rate requirement θ, i.e., $x \geq \theta$. Then $r(x)$ is given as,

$$r(x) = \begin{cases} 1 & if \quad x \geq \theta \\ r_{cos} & if \quad x < \theta, \end{cases} \tag{12}$$

with $r_{cos} > 1$, a design parameter, we will discuss how to select its value in the next section.

For example, it can be easily shown that by applying the above family of optimal congestion control laws to the TCP utility in Eqs. (4) and (5) and let, $z(.) = z_{TCP}(x,t,cg)$, where,

$$z_{TCP}(x,t,cg) = \begin{cases} (\alpha+\beta)x \; for \;\; SSP \\ \mu + \beta x \quad for \;\; CAP, \end{cases} \tag{13}$$

then we arrive at the TCP Reno congestion control law [27].

Now applying the above family of optimal congestion control laws to the NUM problem given in Eq. (6), (i.e., maximizing the total user utility for users with utility function $U(x) = wU_{TCP}(x)$), we arrive at PACCP as follows. In the SSP,

$$\dot{x} = \begin{cases} \left(3r(x) - 3^{1-\omega}\right)\alpha x/2 & if \quad cg = 0 \\ -3^{-\omega+1}\beta x & if \quad cg = 1, \end{cases} \tag{14}$$

and in the CAP,

$$\dot{x} = \begin{cases} \left[-\left(\frac{\beta x}{\mu+\beta x}\right)^{\omega} + r(x)\right](\mu+\beta x) & if \; cg = 0 \\ -\left(\frac{\beta x}{\mu+\beta x}\right)^{\omega}(\mu+\beta x) & if \; cg = 1, \end{cases} \tag{15}$$

Clearly, flow rate allocation is determined by $r(x)$ (or θ) and ω, a pair of parameters that uniquely determine PACCP. Equations (14) and (15) are a family of congestion control laws including TCP Reno with $\omega = 1$ and $\theta = 0$ (i.e. $r(x) = 1$) and MRG [27] with $\omega = 1$ and $\theta \neq 0$ as their special cases.

To be backward compatible with TCP window-based congestion control, we translate the fluid-flow based control laws in Eqs. (14) and (15) into window-based congestion control protocols. In the window-based control, the flow rate stays unchanged during each RTT interval. Hence, the congestion window can be calculated as $W_c = xRTT/MSS$, where MSS is the maximum segment size. The flow rate change from one RTT epoch to the next RTT epoch is $\Delta x = \dot{x}RTT$, where $\dot{x}$ is the flow rate change over an RTT epoch, which can be estimated by the fluid-flow control law. Hence, the window size change ΔW_c at the end of every RTT epoch is calculated as $\Delta W_c = \Delta xRTT/MSS$. The minimum congestion window size W_{min} corresponding to a minimum guaranteed rate θ is given as follows,

$$W_{min} = \frac{\theta RTT}{MSS}. \tag{16}$$

For a flow without minimum rate requirement, $W_{min} = 0$ (i.e., θ=0).

Define $CD1$ as $\{ cg = 0 \;\&\; W_c < W_{min}\}$ and $CD2$ as $\{cg = 0 \;\&\; W_c \geq W_{min}\}$. Now the window-based protocol for congestion window size (W_c) update (i.e., $W_c = W_c + \Delta W_c$) can be approximated (by assuming $\beta x \gg \mu$) as follows. In the SSP,

$$W_c = \begin{cases} \left(\left(3r_{cos} - 3^{1-\omega}\right)/2 + 1\right) W_c & if \;\; CD1 \\ \left(\left(3 - 3^{1-\omega}\right)/2 + 1\right) W_c & if \;\; CD2 \\ \left(1 - 3^{1-\omega}/2\right) W_c & if \;\; cg = 1, \end{cases} \tag{17}$$

and in the CAP,

$$W_c \approx \begin{cases} \frac{(r_{cos}+1)W_c}{2} + (r_{cos} - 1 + \omega) & if \; CD1 \\ W_c + \omega & if \; CD2 \\ \frac{1}{2}W_c + (\omega - 1) & if \; cg = 1. \end{cases} \tag{18}$$

Note that TCP Reno is a special case of the above control protocol with $r_{cos} = 1$ (i.e., $\theta = 0$) and $\omega = 1$. A flow under the control of the above PACCP receives minimum rate guarantee and quantifiable fair sharing of the additional bandwidth. PACCP supports three broad CoSes based on specific settings of the pair of parameters (θ, ω), i.e., the best effort (BE) service with (0, 1) (i.e., TCP Reno); the differentiated service (DS) with (0, ω >1); and the minimum rate guaranteed (MRG) service with (θ >0,

$\omega \geq 1$). The three CoSes can be enabled by simply passing a pair of control parameters, i.e., (θ, ω), into PACCP. Hence, pricing models tied to this pair of control parameters may be developed to charge users in proportion to the (relative) network bandwidths allocated to them. For datacenter with both TCP and UDP traffic, the two parameters can also be enabled to DCCP congestion control protocols [31, 32] to provide the three different types of services for UDT traffic. The only difference is that there is no retransmission in UDP traffic in case of congestions.

Assigning different weights to different flows is applied in Seawall [47] and NetShare [48] to allocate the flow rate to provide max-min fairness. However, unlike PACCP, the flow weight used in Seawall and NetShare is not considered in congestion control, and hence making them difficult to achieve work conservation (i.e., high network utilization). Moreover, PACCP is an optimal solution maximizing the total user utilities while Seawall and NetShare are empirical design.

Pricing model

As different applications may have different flow rate demands. An application can buy a type of service for its generated flows by paying a corresponding price. For example, an application requiring a minimum rate can pay a price to buy a minimum guaranteed rate θ; and an application without minimum guaranteed rate, θ, is set to be 0, but such flows with larger utility weight ω can obtain more bandwidth when they compete the bandwidth from each other. Hence a flow with a larger ω needs to pay more than a flow with a smaller ω. In the following subsections, we discuss how to set the pair of parameters based on the flow prices in PACCP to support the three CoSes. Two pricing models, a coarse-grained VM-Based Pricing model (VBP) and a fine-grained Flow-Based Pricing model (FBP), are proposed to support the three CoSes.

In VBP, a user paying for the usage of a VM instance will also pay a network usage fee per unit time for an aggregated bandwidth determined by a given pair, (θ, ω). In this model, in principle, each VM instance can support more than one CoS, as long as, $\sum_i^{n_v} \theta_i \leq \theta$ and $\sum_i^{n_v} \omega_i \leq w$ (it can be easily shown that both parameters are additive), where n_v is the number of active flows emitted from the instance and (θ_i, ω_i) is the pair of control parameter for flow i, for $i = 1, ..., n_v$. Namely, the only requirement is that the network bandwidth taken by all the flows emitted from this instance is upper bounded by the aggregated bandwidth allocated to the instance. However, as VBP is meant to be design as a coarse-grained, easily implementable pricing model, we limit the scope of VBP to the case where each VM instance only support a single CoS, whether it is BE, DS, or MRG. Moreover, all the flows emitted from the instance gain equal share of network bandwidth, i.e., $(\theta_i, \omega_i) = (\theta/n_v, w/n_v)$, $\forall i$. VBP is a static pricing model, allowing the network bandwidth to be purchased as an integral part of a VM instance. However, a major drawback of VBP is that the aggregated bandwidth is statically pre-allocated and cannot be quickly adjusted to respond to network resource demand changes.

To address the above drawback of VBP, we also propose FBP. In this model, a customer pays an initial purchase fee for the default BE CoS as an integral part of a VM instance and then pays the DS and MRG CoSes on a per-flow-basis on demand. It also allows dynamic runtime service upgrading or downgrading by changing the pair of parameters and the corresponding price. FBP is more flexible than VBP, but is harder to implement and manage. In what follows, we propose pricing structures for the two models, separately.

VBP: VM-Based pricing model

We propose to use the following pricing structure for VBP corresponding to the three CoSes.

BE CoS: This is the default CoS with (θ, ω)=(0, 1). The price, P_{BE}, for this CoS may be set at, $P_{BE} = P_0$, where P_0 is a fixed basic price per unit time.

DS CoS: For this CoS, $(\theta, w) = (0, w > 1)$. The price, P_{DS}, for this CoS can be modeled as $P_{DS} = P_0 + P_1(w-1)$, where P_1 is a price per unit time. As we will show in the next section, a DS flow with $w > 1$ usually to be consistently smaller than the optimal one (i.e., ω times of the BE flow rate)(will be explained later). All our results suggest that the average rate of DS flows with ω=2 is about 1.6 to 1.7 times of the BE flow rate at high network load, and hence P_1 may be set at $0.6P_0$ to ensure that the flow rate is indeed proportional to the price paid.

MRG CoS: The price, P_{MRG}, for the MRG CoS can be formulated as $P_{MRG} = P_0 + P_1(w-1) + P_2\theta$, where P_2 is a price per unit data, in association with the minimum guaranteed rate.

Clearly, with VBP, in addition to CPU speed and memory size, a CoS with a specific (θ, w) pair can now be included for price tagging a VM instance. For example, a user may want to purchase VM instances with MRG CoS. Based on the application characteristics and an expected number of concurrently active flows, VM instances with a pair of parameters (θ, w) that matches the demand may be purchased, and making the performances of VM instances proportional to their prices paid.

FBP: flow-Based pricing model

In FBP, a customer is charged upfront for the use of the BE CoS. However, to use DS or MRG CoS, the customer will incur an additional charge, according to the specific values of the pair of parameters (θ, ω) chosen for the flow. The additional charge, P^s, may follow a similar pricing structure as for the MRG CoS in VBP, i.e., $P^s = P_0^s + P_1^s(w-1) + P_2^s\theta$. Here P_0^s, P_1^s and P^s are the

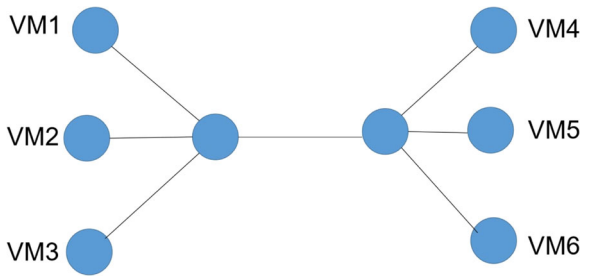

Fig. 1 An example network

price per unit time and P_2^s are the price per unit of data sent.

Since our focus in this paper is on the price versus performance consistency, how to determine the parameters in the pricing structures, i.e., $P_0(P_0^s)$, $P_1(P_1^s)$, $P_2(P_2^s)$, to balance the profit and user satisfaction is subject to future investigation.

Rate allocation examples

Now we use a simple network topology presented in Fig. 1[3] to illustrate how PACCP allocates flow rates under the two pricing models. Data flows are sent between three source-destination pairs, i.e., {VM$_i$, VM$_{i+3}$} (i= 1, 2 and 3), through a shared bottleneck link with bandwidth B. Assume that the propagation delays are the same for all the flows. In the follows, we discuss how the rates are allocated by applying VBP model and FBP model, respectively.

VBP: Assume that VM$_i$ (i= 1, 2 and 3) has n_i flows sending to VM$_{i+3}$, respectively. We first examine the case where all flows are of BE CoS. In this case, the optimal rate allocation is such that each flow emitted from VM$_i$ receives $B/3n_i$ allocated bandwidth, respectively. Specifically, the shared bandwidth is first equally allocated to the three VMs with $B/3$ each, which is further equally allocated to each flow in a VM. For example, let n_1=1, n_2=2, and n_3=3. Then a BE flow from VM1, VM2, or VM3 is allocated $B/3$, $B/6$ or $B/9$ bandwidth, respectively.

Now we consider the case where there are two types of VMs running either BE or DS flows. Specifically, assume that both VM1 and VM2 run BE flows, and VM3 has DS flows with the pair of parameters (0, ω). In this case, the optimal rate allocation is such that each BE flow from VM1 and VM2 is allocated the bandwidth of $B/((2+\omega)n_1)$ and $B/((2+\omega)n_2)$, respectively, and each DS flow gets $\omega B/((2+\omega)n_3)$. For example, suppose that n_1=1, n_2=n_3=2 and ω=2, then each BE flow from VM1 and VM2 is allocated $B/4$ and $B/8$, respectively, and each DS flow from VM3 gets $B/4$.

[3]This topology is different from the leaf-spine datacenter topology. However, if a datacenter has single bottleneck link at a time, the leaf-spine topology can be modeled as this topology.

Finally we exam the case with the presence of all three CoSes. Specifically, assume that VM1, VM2, and VM3 have BE flows, DS flows with $(0, \omega_1)$, and MRG flows with (θ, ω_2), respectively. The optimal rate allocation is then to first allocate θ rate to VM3, and then proportionally allocates the remaining bandwidth to the three VMs to maximize the total utility. If θ =0, VM3 would be allocated a bandwidth of $B\omega_2/(1+\omega_1+\omega_2)$. Otherwise, it would receive at least θ. As a result, VM3 gets $B_3 = max\{\theta, B\omega_2/(1+\omega_1+\omega_2)\}$. Hence each MRG flow gets B_3/n_3, assuming that the minimum guaranteed rate is evenly assigned to each MRG flow. Then the remaining bandwidth $B_{BD} = B - B_3$ are allocated to VM1 and VM2, and hence VM1 and VM2 receive, $B_1 = B_{BD}/(1+\omega_1)$ and $B_2 = \omega_1 B_{BD}/(1+\omega_1)$, respectively, with each BE flow gets bandwidth B_1/n_1 and each DS flow gets B_2/n_2.

Let look at an example by assuming that n_1=1, n_2=n_3=2, $\omega_1 = 2$ and $\omega_2 = 1$. We first let $\theta = B/2$. In this case, the optimal flow rates allocated to the three VMs are $B/6$, $B/3$ and $B/2$, respectively. The optimal rate is $B/6$ for a BE flow in VM1, and $B/6$ for a DS flow in VM2 and $B/4$ for a MRG flow in VM3. Now assume that $\theta = B/10$, then the optimal rate allocations to the three VMs are $B/4$, $B/2$ and $B/4$, respectively. Hence each BE flow in VM1 has $B/4$ bandwidth, and each DS flow in VM2 has $B/4$ bandwidth and each MRG flow in VM3 has $B/8$ bandwidth, respectively.

FBP : For this model, the optimal flow rate allocation for each flow is independent of the VM instance the flow is originated from. More specifically, assume that there are n_i^{BE}, n_i^{DS} and n_i^{MRG} BE, DS and MRG flows emitted from VM$_i$ (i= 1, 2, and 3). Also assume that the pairs of parameters for all the flows belonging to the same CoS are the same. Namely, for BE, DS and MRG flows, they are (0,1), $(0, \omega_1)$ and (θ, ω_2), respectively. Also let, $n^{BE} = \sum_{i=1}^{3} n_i^{BE}$, $n^{DS} = \sum_{i=1}^{3} n_i^{DS}$ and $n^{MRG} = \sum_{i=1}^{3} n_i^{MRG}$. Then, the optimal flow rate allocation for flows from different VMs is dependent on n^{BE}, n^{DS}, n^{MRG} and CoSes only, not from which VMs they come from.

The optimal rate allocation is to first satisfy the minimum rate, θ, for all n^{MRG} MRG flows, and then allocates the remaining bandwidth to BE, DS and MRG flows in proportional to their weight values. Specifically, an MRG flow gets B_{MRG}=$max(\theta, B\omega_2/(n^{BE} + \omega_1 n^{DS} + \omega_2 n^{MRG}))$. Then the remaining bandwidth $B_{BD} = B - n^{MRG} B_{MRG}$ is allocated to BE and DS flows with each BE flow getting $B_{BD}/(n^{BE} + \omega_1 n^{DS})$ and each DS flow getting $\omega_1 B_{BD}/(n^{BE} + \omega_1 n^{DS})$.

For example, assume that n_i^{BE}=n_i^{DS}=n_i^{MRG}=1 (i= 1, 2 and 3) and $\omega_1 = \omega_2 = 2$. We first assume that $\theta = B/6$. In this case, $B_{MRG} = max(B/6, 2B/15) = B/6$, and then $B_{BD} = B/2$, so the optimal rate allocation is $B/18$ for a BE flow, $B/9$ for a DS flow and $B/6$ for a MRG flow. Now assume that $\theta = B/20$. In this case, $B_{MRG} = max(B/20, 2B/15) =$

$2B/15$, then $B_{BD} = 3B/5$. So the optimal rate allocations are $B/15$ for a BE flow, $2B/15$ for a DS flow and $2B/15$ for a MRG flow.

The power of PACCP lies in the fact that with the right assignment of the pair (θ, w) in PACCP for each flow, the congestion control is automatically enabling users' prices into rate allocation and leads to the optimal price-aware rate allocation for any network topology without bandwidth reservation. Hence PACCP is readily to be implemented in today's datacenters for charging the network resource usage.

Implementation issue

PACCP is a fully distributed price-aware congestion control protocol. To implement PACCP, it only needs to update the congestion control software on the sending side hosts without any core network involved. The only modification in the sending side host is to adjust the rate increase/decrease using the pair of parameters (θ, ω) which are price oriented parameters. The complexity of PACCP is low as it only needs to do a simple calculation based on the current window size and the minimum congestion window size W_{min}, i.e, if the current window size is greater than W_{min} or not.

PACCP can be easily implemented in Linux kernel on the sending hosts. The pair of parameters (θ, ω) can be passed from the user space to the Kernel space through some standard device control functions, e.g., *ioctl*(). PACCP can also be implemented in other operating systems (OS) such as Windows, the pair of parameters can be set through configuration registry using command *Tcpip*. If an OS is managed by the cloud service provider, the price charge can be directly executed by setting up/monitoring the pair of parameters passed from the user space to the Linux kernel. If an OS is administrated by a tenant, the cloud service providers can move the congestion control from a data path to a congestion control plane [33, 34] or a virtual switch [35] which can be implemented in the network interface cards, and the price is charged based on the pair of parameters used in the congestion controllers.

The scalar parameter selection

The parameter r_{cos} is related to the minimum guaranteed flow rate in PACCP. $r_{cos} > 1$ for a flow with a minimum guaranteed rate $\theta > 0$. The value of r_{cos} determines the speed for a flow to reach its minimum guaranteed rate. A larger r_{cos} value allows a flow to reach its minimum guaranteed rate quickly. However, aggressive flow rate acceleration may cause more congestion and larger flow rate oscillations, and more negative impact on overall system performance. Hence, r_{cos} must be selected to strike a balance between acceleration speed and congestion control. In this subsection, we find the desired range of r_{cos} through simulation.

This study is carried out based on an event-drive simulator written in C++. A 6x5 leaf-spine network topology with each rack having 40 hosts[4] is used. The bandwidth/propagation delay is set at 10Gbps /10 μs between a host and a leaf node and 40Gbps/20 μs between a leaf node and a spine node. The queue sizes for the 10/40 Gbps links are set at 150/450 Kbytes (i.e., 100/400 packets).

In datacenters, the flow size varies significantly [16, 23, 36]. While most of the flows are short-lived, having flow size less than 100K, a small percentage of long-lived, big flows consume most of the network bandwidth. Hence we apply the real datacenter workloads, i.e., Data-mining [36] and Websearch [16], as input to study the parameter selection of r_{cos}. In the simulation, the flows are dynamically generated, following a Poisson process. The average flow arrival interval is used as a tunning knob to set the network load at desired levels. When a flow arrives, a source host is randomly selected and then a destination host is randomly selected from a different rack. We consider a system with mixed BE and MRG flows (with $\omega = 1$). Suppose that in each rack, there are 20 hosts running BE and MRG flows, respectively. The flows are classified into small (size<=100 Kbytes), medium (100< *size* <=1 Mbytes) and big (size> 1 Mbytes) flows. The deadlines for MRG flows are set at 0.7ms, 2ms and 3Gbps (i.e., deadline=flow size/3Gbps) for small, medium and big flows, respectively. We consider three performance metrics, i.e., the flow deadline miss rate, average flow completion time (FCT) and 99th percentile of FCT (including both BE and MRG flows).

Figure 2 gives the results under three network loads (i.e., 30%, 50% and 70%) in Data-mining workload. We see that a value of $r_{cos} > 1$ (note that the BE and MRG flows are the same with $r_{cos} = 1$) can significantly reduce the flow deadline miss rate. The flow deadline miss rate decreases for all network loads, as r_{cos} increases up to 3, and it hardly further reduces, as r_{cos} goes over 3. The average FCT at light and medium loads (i.e., 30% and 50%) decreases as r_{cos} increases from 1 to 3, and then increases as r_{cos} further increases. The 99th FCT at heavy load (70%) increases as r_{cos} increases.

Figure 3 shows the results in Websearch workload. The results are similar to these in Data-mining workloads. Hence we conclude that r_{cos} around 3 (i.e., from 2.5 to 3.5) can achieve the overall best performance for the systems with mixed BE and MRG flows. In the rest of the case studies in this paper, we set r_{cos}=3.

Performance evaluation

In this section, we first examine the optimality of PACCP by simulation as well as a small testbed implementation, and then test the price-performance consistency of

[4] In datacenters, a host can host one or multiple VMs. For simplicity, we assume that each host runs a single VM. Hence, VM and host are used exchangeably in the rest of the paper.

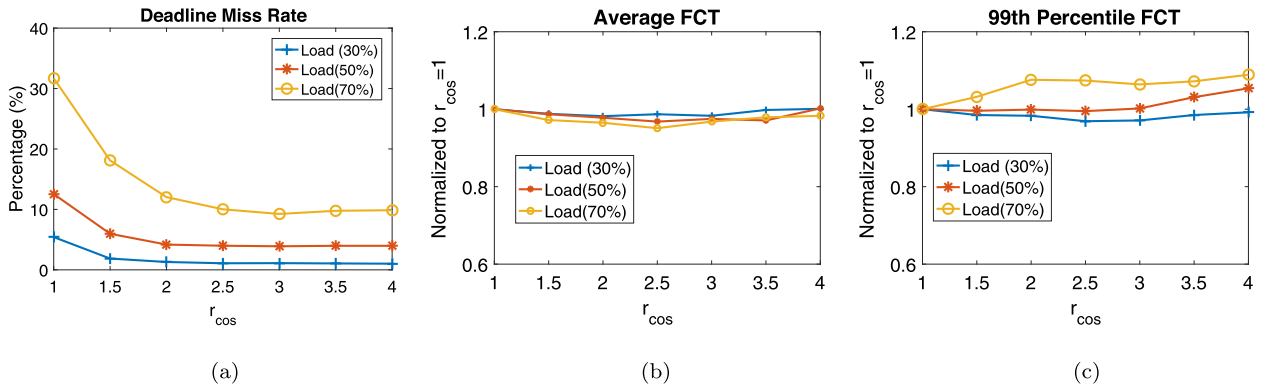

Fig. 2 Impact of r_{cos} (Data-ming workload)

PACCP for both pricing models in a large datacenter based on real-world workloads.

Optimality test by simulation

We first test PACCP in terms of the user utility maximization and optimal rate allocation by simulation. We use the same datacenter topology, i.e., a 6x5 leaf-spine network with 40 hosts in each rack. We first consider a simple case, i.e., each host has one outgoing flow and one incoming flow. So there are a total number of 40 outgoing flows (20 BE, 10 DS and 10 MRG flows) in each rack, with 8 of them (i.e., 4 BE, 2 DS, and 2 MRG flows) going to exactly one of the other 5 racks. To test the optimality of PACCP, we assume that all the flows are extremely long-lived with unlimited amount of data to send. With this setup, 40 flows from each rack share a total of 200 Gbps (i.e., five 40 Gbps links) outgoing bandwidth. We first set the pairs of parameters to be (0, 1), (0, 2) and (2Gbps, 1) for BE, DS and MRG flows, respectively. In this case, the optimal flow rate allocation for each 40 Gbps leaf-spine link are 4 Gbps, 8 Gbps and 4 Gbps for each of the 4 BE, 2 DS and 2 MRG flows, respectively. We also consider another case where the only difference from the previous case is that the pair of parameters for MRG flows is changed to (7Gbps, 2). In this case, the optimal flow rates for each of the BE, DS and MRG flows are 3.25, 6.5 and 7 Gbps, respectively. Since for both cases, each VM only sources one flow, the flow rate allocations are the same for both VBP and FBP.

Figures 4 (a) and (c) show the sum of user utility from simulation and the optimal one, V in Eq. 6, normalized to one. As we can see, the simulated one closely matches with, but is slightly lower than the optimal one for both cases. The reason why it is always lower than the optimal one is that for any transport congestion control protocol including PACCP, the aggregate flow rate cannot achieve 100% link utilization all the time, due to congestion feedback delay and finite granular control.

The rates of the three CoS flows, each averaged over all the flows in the same CoS, are depicted in Figs. 4 (b) and (d). The average flow rates over time for BE, DS and MRG flows are 3.72/6.21/4.09 Gbps and 2.98/5.08/7.14 Gbps for the cases of θ =2 Gbps and 7 Gbps, respectively. The results indicate that the rates of MRG flows are always above the minimum guaranteed rate θ. The rate ratio between DS and BE is about 1.67/1.71, smaller than the optimal ratio 2, for both cases. This is because the optimal ones are obtained based on the assumption that the PACCP controllers for both BE and DS flows shar-

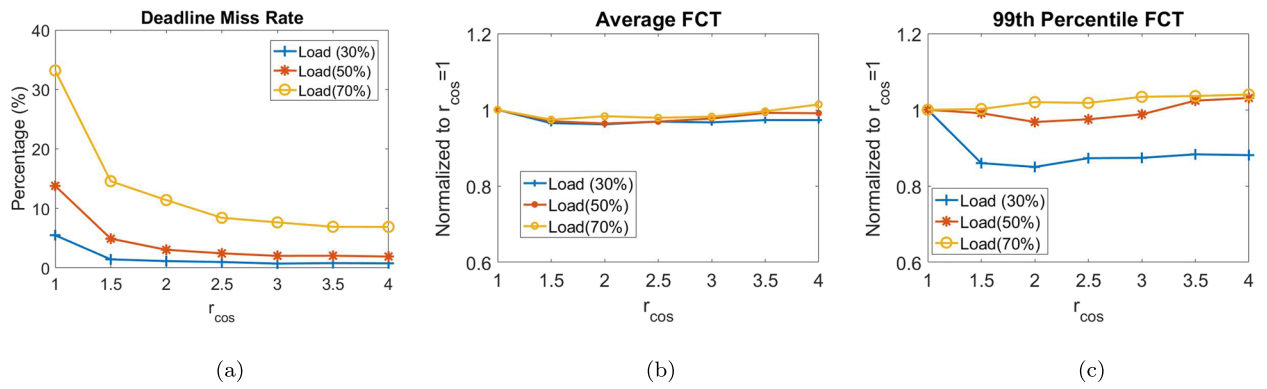

Fig. 3 Impact of r_{cos} (Websearch workload)

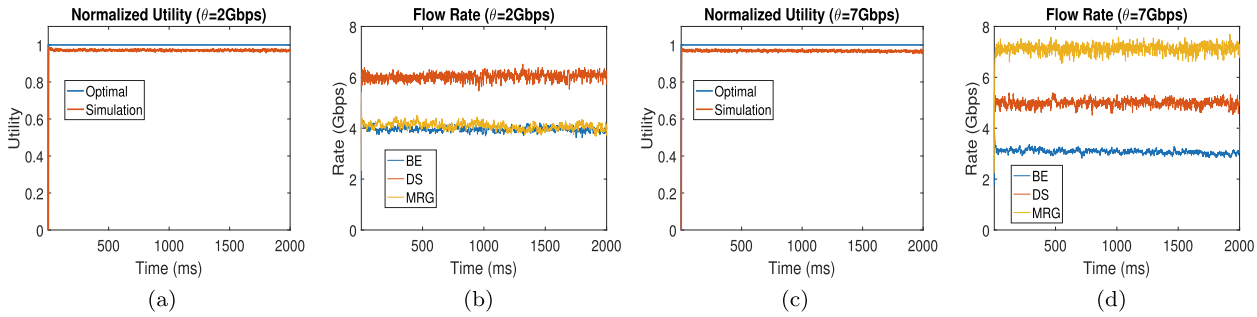

Fig. 4 Normalized utility and rate allocation for each host having one flow: (**a**) and (**b**) θ =2 Gbps; (**c**) and (**d**) θ =7 Gbps

ing the same bottleneck links will sense the congestion simultaneously. In practice, however, a flow of higher rate may sense more congestions than a flow of lower rate, and hence DS flows will incur more rate reduction. This explains why the flow rate ratio of the DS and BE flows is less than the optimal one.

To further test VBP, we create two types of hosts by adding one additional outgoing flow to each of the 10 BE, 5 DS and 5 MRG hosts in each rack, forming a second type of hosts, leaving the other half of hosts in each rack unchanged. As a result, each of this type of hosts now has 2 outgoing flows. The flows generated from this type of hosts are called BE-2, DS-2 and MRG-2 flows, and the flows generated from the other hosts are denoted as BE-1, DS-1 and MRG-1 flows.

Now we consider the pairs of parameters (0, 1), (0, 2) and (7 Gbps, 2) for BE, DS and MRG hosts, respectively. This means that the pairs of parameters for BE-1, DS-1 and MRG-1 flows are (0,1), (0,2) and (7Gbps, 2), respectively, and the pairs of parameters for BE-2, DS-2 and MRG-2 flows are (0, 0.5), (0,1) and (3.5 Gbps, 1), respectively. As a result, the optimal flow rate allocation is 3.25 Gbps, 6.5 Gbps and 7 Gbps for BE-1, DS-1 and MRG-1 flows, respectively, and 1.625 Gps, 3.25 Gps and 3.5 Gbps for BE-2, DS-2 and MRG-2 flows, respectively.

Figure 5 (a) shows the results for the normalized user utility. Again, the simulated one is very close to the optimal one. Figure 5 (b) presents the simulated flow rates of individual types and CoSes. The average flow rates for BE-1/2, DS-1/2 and MRG1/2 are found to be 2.74/1.6, 4.65/2.66 and 7.16/3.85, respectively. Similar to the previous case, the rates of MRG flows are always above their required minimum rates and the flow rate ratios between DS-1 and BE-1, and DS-2 and BE-2 flows are about 1.66 and 1.7, lower than the optimal value of 2. The flow rate ratio between BE-1 and BE-2, DS-1 and DS-2, and MRG-1 and MRG-2 are 1.65, 1.78 and 1.86, respectively, also lower than the optimal value of 2, for the same reason explained earlier.

For the current case and under VBP, flows of the same CoS and from different VMs should be allocated the same total rate. For example, a DS-1 flow originated from one host should receive the same flow rate as the sum of two DS-2 flows originated from another host. Figure 5 (c) shows the testing results for the average flow rates originated from different types of hosts. We can see that the average flow rates from the two types of hosts of the same CoS are reasonably close to each other, with the rates from type 2 slightly higher than that of type 1. This is caused by the fact that each of the two flow from a type 2 host has a smaller flow rate than that of a flow from a type 1 host, and hence they sense less congestion signals, as explained earlier.

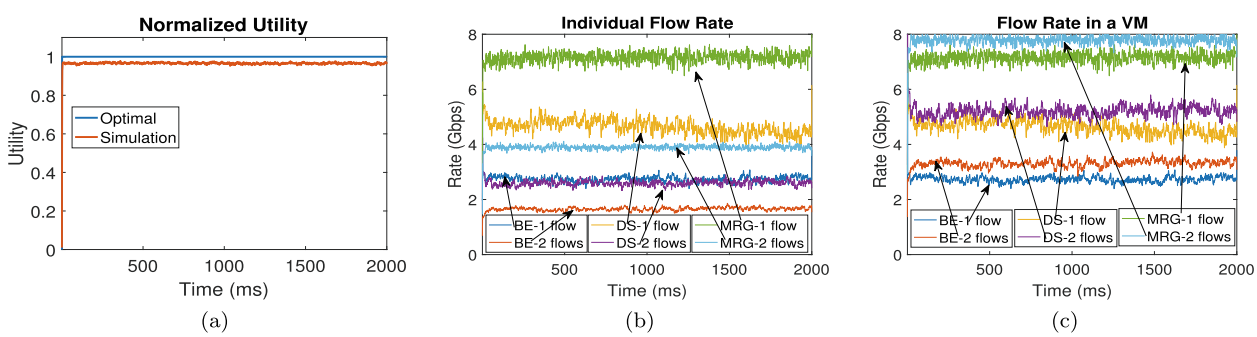

Fig. 5 Normalized utility and rate allocation for half hosts having one flow and half hosts having two flows : (**a**) Normalized utility; (**b**) Average flow rate; and (**c**) Total per VM rate

Optimality test in a real testbed

We implement our PACCP in the Linux kernel. Our Linux code is modified based on the TCP Reno. In PACCP, the minimum guaranteed rate θ and the flow utility weight ω are passed from the user space to the Kernel space through the standard device control function, *ioctl*(). With the pair of values θ and ω, a minimum window size W_{min} is calculated using θ and the measured in RTT (which is already measured in TCP Reno). The only difference between PACCP and TCP Reno is that PACCP needs firstly to compare the current congestion window size with W_{min}, and then compute the adjusted congestion window using Eqs. (17) and (18) for each received acknowledgement.

A 3x1 leaf-spine (i.e., 3 leaf nodes and 1 spine node) datacenter network topology as shown in Fig. 6 is set up using four Dell N4032F switches. Each link has 1 Gbps bandwidth. VMs 1, 4 and 7 are BE hosts, VMs 2, 5 and 8 are DS hosts and VMs 3, 6 and 9 are MRG hosts. Each VM originates 1 long-lived flow. Hence the flow rate allocation is the same, regardless whether VBP or FBP is in use. The destinations of VM i are $(i + 3)\%9$ (for i=1, 2, ..., 9). The pairs of parameters are set at (0,1), (0, 2) and (400Mbps, 2) for BE, DS and MRG flows, respectively. With this setup, the optimal flow rates are 200 Mbps, 400 Mbps and 400 Mbps for BE, DS and MRG flows, respectively.

Figure 7 shows the average flow rates for flows of the three CoSes. The average rate of MRG flows is about 410 Mbps, above the minimum guaranteed rate 400 Mbps. The average rate of DS and BE flows are about 310 Mbps and 180 Mbps, respectively, resulting in a flow ratio of about 1.7, less than the optimal ratio 2. These results agree with the simulation results.

In summary, both simulation and testbed testing results indicate that with PACCP, MRG flows have high chance to meet their minimum guaranteed rates. Moreover, the DS flows can indeed gain more bandwidths when they compete with BE flows, which however, are consistently lower than the optimal ones. This implies that the pricing parameters for DS flows in both VBP and FBP need to be adjusted to reflect the biased relative flow rate. We found that a DS flow with ω=2 achieves about 1.6

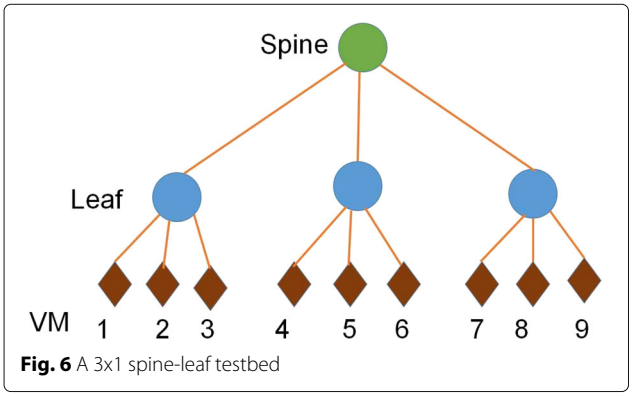

Fig. 6 A 3x1 spine-leaf testbed

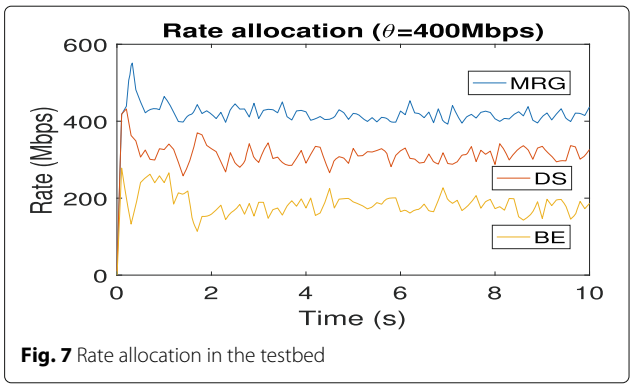

Fig. 7 Rate allocation in the testbed

times the flow rate as a BE flow at high load. Although more detailed study at higher weight values are warranted, this observation does suggest that one may set P_1 (P_1^s) at some smaller value than P_0(P_0^s), e.g., 0.6 P_0 (P_0^s) for VBP (FBP). Note that DS is meant to outperform BE at high load. At low load, they offer similar performance. Hence, DS related pricing must be determined at the high load.

The both simulation and real testbed test results show that PACCP can closely achieve the expected theoretical rate allocation. Hence the pair of parameters (θ, ω) can be real effective to map the flow prices.

Performance evaluation with real datacenter workloads

Now the flow allocations using PACCP with the two pricing models are tested using real datacenter traffic workloads, i.e., Websearch [16] and Data-mining [36], as input. The focus is placed on the testing of the price-performance consistency, i.e., whether the flow rate allocated by PACCP matches the expected rate allocation based on the two pricing models.

Again we use the same network setup as the previous one, i.e., a 6x5 leaf-spine network topology with each rack having 40 hosts. The overall FCT and FCTs for small, medium and big flows are measured as performance metrics. The overall DMR and DMRs for small, medium and big flows for flows with deadlines are also measured as the performance metrics. Although BE and DS flows are deadline unaware, we compare the DMRs for flows with deadline using the BE and DS services against that using the MRG service to reveal how much MRG can help improve DMR over the other two.

VBP

We first examine the performance of PACCP with VBP. Consider the case where there are 20, 10 and 10 hosts in each rack running BE, DS and MRG flows, respectively. The pairs of parameters are set at (0, 1), (0,2) and (5 Gbps, 2) for BE, DS and MRG flows, respectively. The flow deadline for each of n_a active outgoing MRG flows at a host is set at the flow size divided by $5/n_a$ Gbps. We assume that

the flow deadline is lower bounded at 1 ms, as the PACCP connection setup time is taken into account.

We first run PACCP using the Websearch workload [16]. Figures 8 (a) and (b) present the average FCTs for the overall, small, medium and big flows (normalized to the FCT for the BE flows). We see that both DS and MRG flows indeed perform better than BE flows for all load cases. For small and medium DS/MRG flows, their FCTs are less than 0.8 times (i.e., the flow rates are 1.25 times higher) of BE flows at all load cases. As these flows are short-lived flows and may be completed before they reach their optimal allocated flow rate, the performance gains for these flows come from the faster rate increase with the help of ω and r_{cos} (i.e., θ). The results indicate that PACCP is really effective for short-lived flows to enabling price based rate allocation. At light loads, the difference for big flows is small, about 0.9 times of that for the BE flows. This is because at light loads, there is enough bandwidth to accommodate the desired user utilities for all the individual flows of different CoSes, which hardly need to compete against one another for the network resource. Hence long flows have enough time to fully explore the available bandwidth, making the small performance gains.

The performance gains increase quickly as the network load increases. In the high load region (e.g., at 80%), the FCTs for the overall/small/medium/big DS and MRG flows are about 0.62/0.62/0.6/0.63 and 0.61/0.62/0.58/0.62 times of BE flows, respectively. In other words, the average flow rate allocated to DS flows versus BE flows is about 1.6 and 1.7 times, lower than the optimal ratio of 2, which agrees with the findings for the previous long-lived flow cases. MRG flows perform slightly better than DS flows for all cases. The performance gains are about 5% for small and medium flows, but very little for the overall and big flows. The close performance between the DS and MRG flows arises because both DS and MRG have the same weight value of 2. Hence, they are expected to receive equal flow rate allocation, provided that the minimum guaranteed rates for MRG flows are satisfied, which is indeed the case. The fact that MRG flows perform slightly better is because MRG flows open up their send windows faster than DS flows until the flow rates reach their minimum guaranteed rates.

Figures 8 (c) and (d) show the DMRs for the overall, small, medium and big flows. The DMRs for MRG flows are always higher than BE and DS flows. The overall DMR for MRG flows is above 90% even at high load, whereas the DMR for BE flows is below 50%. While the DMRs for medium and big MRG flows are above 90%, the corresponding DMRs for DS and BE flows drop below 80% and 50%, respectively. This clearly demonstrates the importance of MRG in providing high probability of meeting flow deadlines.

Now let us examine the performance of PACCP with VBP using the Data-mining workload [36] as input. Figure 9 depicts the results, which are similar to those for the Websearch workload case. However, we note that the overall FCT performance gains are reduced. The FCT gain for small flow is almost a constant for all load cases for the following reason. Most of the small flows in the Data-ming workload are composed of only a few packets (about 40% flows have a single packet and about 80% flows have no more than 6 packets), which finish in just one or two RTTs, and hence the fast rate increase has limited effect on these flows. The DMRs for the medium and big flows are similar to those for the Websearch case. However, the DMRs for small MRG, DS, and BE flows are high, above 98%, 95% and 90%, respectively, due to the high chance for small flows to finish within the 1 ms deadline.

The above results further ascertain that PACCP with VBP can indeed provide high probability of providing minimum rate guarantee and allow pricing in proportion to the flow rate allocation.

FBP

Finally, we evaluate the performance of PACCP with FBP. For FBP, a customer can run flows of different CoSes in a VM. In our simulation, an outgoing flow from a host has 60% chance to be a BE flow, 20% chance to be a DS flow,

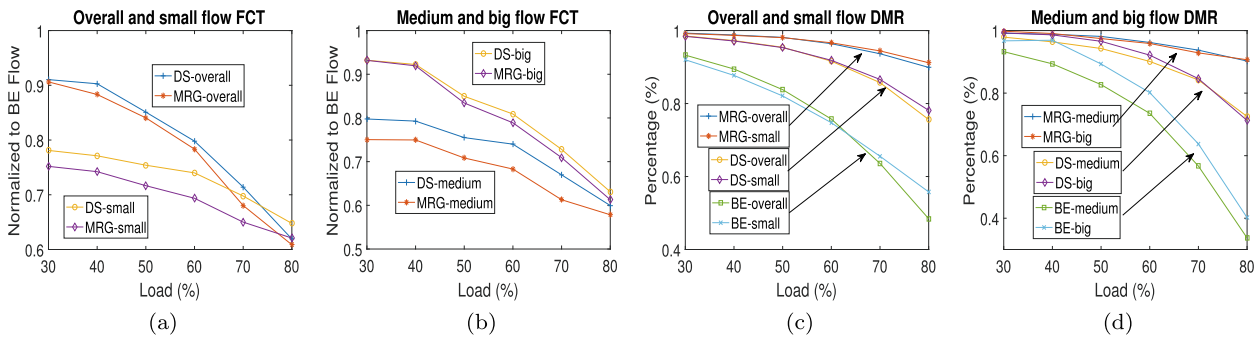

Fig. 8 VBP with Websearch workload: (**a**) overall and small flow FCT; (**b**) Medium and big flow FCT; (**c**) Overall and small flow DMR; and (**d**) Medium and big flow DMR

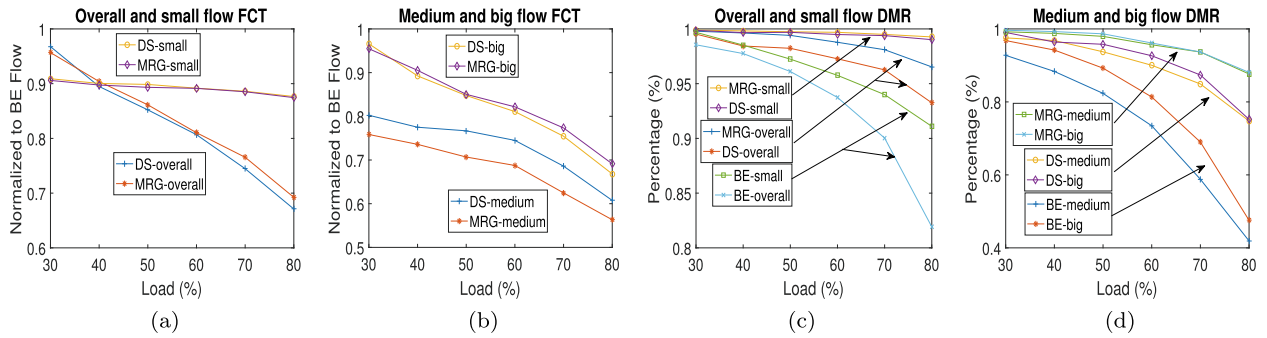

Fig. 9 VBP with Data-ming workload: (**a**) overall and small flow FCTs; (**b**) Medium and big flow FCTs; (**c**) Overall and small flow DMRs; and (**d**) Medium and big flow DMRs

and 20% chance to be an MRG flow. The pair of parameters for BE, DS, and MRG flows are set at (0,1), (0, 2) and (2.5 Gbps, 2), respectively. In FBP, a VM may have all the three types of flows at the same time.

Figures 10 and 11 show the results by using Websearch workload and Data-mining workload, respectively. Overall, the results are similar to these of PACCP with VBP model in both workloads. But the performance gains including both FCTs and DMRs for DS and MRG flows at high loads are slightly less than those in PACCP with VBP model. The utility weight ω in a host in VBP model is the sum of weights from all sending flows, this results in smaller ω used by flows in VBP model. A flow with smaller ω means that its flow increase rate is smaller than that of a flow with a larger ω. Hence the number of congestions in PACCP with ABP model is less than that in FBP with FBP model. As the average flow rate for DS or MRG flows is much higher than that for BE flow, DS and MRG flows sense more congestions than BE flows do, thus making the performance gains in VBP model are closer to the idea case (i.e., 0.5), and hence greater performance gains than that in PACCP with FBP model. The results indicate that PACCP with both pricing models can provide price performance consistency applied in real datacenters.

The above results demonstrate that the proposed PACCP can indeed enable price-aware flow rate allocation in cloud, particularly for short-lived flows, including soft minimum rate guarantee and relative additional rate allocation, commensurate with two pricing models. PACCP is a fully distributed control protocol and only needs software upgrading in the end hosts and does not involve any core switches, and hence it is readily to be implemented in current datacenters.

Related work

As network plays an important role in the performance of cloud services, charging cloud service based on the network performance is necessary and essential. Many research works [1–5, 7, 8, 11–14, 37–41] have studied the network pricing model for cloud services. These pricing models provide bandwidth reservation through an auction process [3, 5, 7, 8, 10–13] or a dynamic time-varying price table [2, 4, 6, 9, 14]. The pricing model based on auction process is a market-oriented pricing strategy and has been heavily studied. The auction mechanisms include game theory approach [5], incentive-compatible auction [7], price-incentive resource auction [13], auction based on the chaotic equation update [3], randomized auction

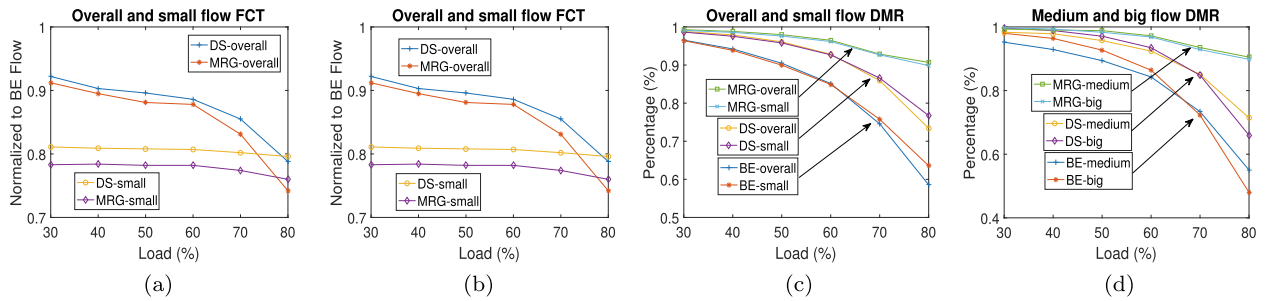

Fig. 10 FBP with Websearch workload (**a**) overall and small flow FCT; (**b**) Medium and big flow FCT; (**c**) Overall and small flow DMR; and (**d**) Medium and big flow DMR

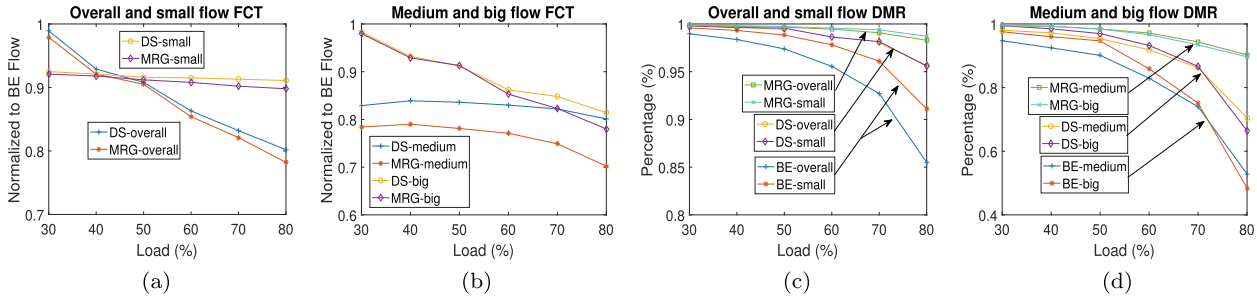

Fig. 11 FBP with Data-ming workload (**a**) overall and small flow FCT; (**b**) Medium and big flow FCT; (**c**) Overall and small flow DMR; and (**d**) Medium and big flow DMR

[11], double auction [10] and deep reinforcement learning [12]. The price in auction approaches is dynamically adjusted based on the market demand and can sufficiently and quickly reflect the economic behavior and hence can benefit both the cloud service providers and tenants. The pricing model based on dynamic varying price table is a pay-as-you-go model, the price is charged based on the real bandwidth usage. It benefits the cloud tenants by preventing their over reserved bandwidths. But the cloud service providers are hard to accurately and quickly set their prices to maximize their business revenue.

Bandwidth reservation can allow tenants to obtain their demanded bandwidths by paying their prices. However, the bandwidth reservation generally incurs significant overheads and the bandwidth allocation can usually be delayed up to seconds, and hence it is not suitable for today's datacenter applications involving bursts of short-lived flows which generally ask the flow completion time in milliseconds. Although a distributed network pricing solution, Softbw [1] can schedule flows at the source hosts and hence can deal with short-lived flows. But, it only works for congestion free datacenter networks.

Although price-agnostic, many solutions have been proposed to improve datacenter flow rate allocation and provide fair bandwidth sharing and/or minimum rate guarantee. Since such solutions may lead to effective pricing models, in what follows, we briefly review some of such solutions.

First, some congestion control protocols with minimum assistance from in-network nodes [16, 18, 42, 43] are proposed to improve the performance of datacenter applications. Some of them [16, 42, 43] are focused on improving average throughput, their congestion controls are based on explicit congestion notification (ECN) or queuing delay, or on the flowlet granularity. D^2TCP provides deadline-aware [18] flow rate allocation through ECN. However they cannot provide provable fairness and minimum flow rate or flow-deadline guarantee and multiple CoSes, as is the case for PACCP. Second, the Hose and Pipe models are widely used for the design of bandwidth allocation schemes [2, 44–49] for datacenter applications. In these schemes, all the VMs are connected to a central (virtual) switch by a dedicated link (hose) for traffic control and minimum bandwidth guarantee. Oktopus [44] and SecondNet [45] support bandwidth guarantee through static reservation. Seawall [47] and NetShare[48] uses flow weight or per-tenant weight for TCP-like flows to achieve max-min fairness. Gatekeeper [50] uses a hypervisor-based mechanism for bandwidth reservation for bisection-bandwidth networks. Tag [49] uses a tenant application graph to more accurately predict the bandwidth demand and hence more effective allocate bandwidth for applications with heterogeneous bandwidth demands. Numfabric [51] is a network utility maximization based weighted fair queuing scheduling to provide proportional fairness. A framework for inter-datacenter traffic pricing model is proposed in [6] to achieve high revenue, but no fairness criteria is enforce.

As the bandwidth reservation is not integrated with the congestion control protocols, these schemes cannot allocate flow rates in a work-conserving manner, hence they may waste network resources. Moreover, the existing schemes are either focus on the bandwidth reservation or the proportional fairness, but rarely on both. The ability to support multiple CoSes and directly enforce flow rate allocation in congestion control makes PACCP a highly resource-effective, work-conserving solution. Moreover, to the best of our knowledge, it is the first price-aware transport congestion control protocol purposely design for cloud applications, and can be readily deployed to current datacenters.

Conclusions

In this paper, we propose PACCP, a price-aware congestion control protocol for cloud services. PACCP is a network utility maximization (NUM) based optimal congestion control protocol and supports multiple CoSes, including best-effort service (BE), differentiated service (DS) and minimum rate guaranteed (MRG) services. PACCP seamlessly integrates congestion control with

two pricing models, a coarse-grained VM-Based Pricing model (VPB) and a fine-grained Flow-Based Pricing model (FBP). The flow rate allocated by PACCP is determined by a pair of parameters, i.e., a minimum guaranteed rate and a utility weight, which are, in turn, determined by the paid price. PACCP is evaluated by both large scale simulation and small testbed implementation. The experimental results demonstrate that PACCP can indeed achieve high probability of providing minimum rate guarantee, high bandwidth utilization and proportional flow rate allocation, commensurate with the two pricing models. PACCP is highly scalable and can deal with burst of unlimted numbers of short-lived flows. It only requires software upgrade in end hosts without any change in core network switches and hence is readily deployable in today's datacenters.

Acknowledgment
We would like to thank participants to UCC conference for their observations.

Authors' contributions
The authors had worked equally during all this paper's stages. All authors read and approved the final manuscript.

Funding
This work was supported by Alibaba Group through Alibaba Innovative Research (AIR) Program, the US NSF under Grant No. CCF XPS-1629625 and CCF SHF-1704504, Science and Technology Projects in Guangzhou No.202102080300 and Guangdong Province Youth Innovation Talent Project No.2017KQNCX108.

Availability of data and materials
All the simulation and Linux implementation codes are available on https://github.com/zjwang68/PACCP.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Computer Science, Guangdong Pharmaceutical University, Guangzhou, China. [2]Medical Information and Real World Engineering Technology Center, Guangdong Pharmaceutical University, Guangzhou, China. [3]Department of Computer Science, The University of Texas at Arlington, Arlington, USA. [4]Future Network Research Center, Purple Mountain Laboratories, Nanjing, China.

Received: 25 March 2021 Accepted: 28 September 2021
Published online: 20 November 2021

References

1. Guo J, Liu F, Wang T (2017) Pricing Intra-datacenter Networks with Over-Committed Bandwidth Guarantee. USENIX ATC, Santa Clara
2. Popa L, Kumar G, howdhury M, Krishnamurthy A, Ratnasamy S, Stoica I (2012) FairCloud: Sharing the Network in Cloud Computing. ACM SIGCOMM, Helsinki
3. Niu D, Feng C, Li B (2012) Pricing cloud bandwidth reservations under demand uncertainty. ACM SIGMETRICS Perform Eval Rev 40(1):151–162
4. Ballani H, Jang K, Karagiannis T, Kim C, Gunawardenam D, O'Shea G (2013) Chatty tenants and the cloud network sharing problem. Usenix NSDI, Berkeley
5. Shen H, Li Z (2014) New Bandwidth Sharing and Pricing Policies to Achieve A Win-Win Situation for Cloud Provider and Tenants. IEEE INFOCOM, Toronto
6. V Jalaparti V, Bliznets I, Kandula S (2016) Dynamic pricing and traffic engineering for timely inter-datacenter transfers. ACM SIGCOMM, Florianopolis
7. Jin A, Song W, Wang P, Niyato D, Ju P (2016) Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. IEEE Trans Serv Comput 9(6):895–909
8. Jin A, Song W, Zhuang W (2018) Auction-based resource allocation for sharing cloudlets in mobile cloud computing. IEEE Trans Emerg Top Comput 6(1):45–57
9. Kansal S, Kumar H, Kaushal S, Sangaiah A. K (2020) Genetic algorithm-based cost minimization pricing model for on-demand IaaS cloud service. J Supercomput 76:1536–1561
10. Dibaj R, Miri A, Mostafavi S (2020) A cloud priority-based dynamic online double auction mechanism (PB-DODAM). J Cloud Comput 9(64):1–26
11. Le T, Tran N, Leanh T, Kim T, Ren S, Hong C (2020) Auction mechanism for dynamic bandwidth allocation in multitenant edge computing. IEEE Trans Veh Technol 69(12):162–176
12. Shi B, Huang L, Shi R (2020) Pricing in the competing auction based cloud market: A multi-agent deep deterministic policy gradient approach. In: Kafeza E., Benatallah B., Martinelli F., Hacid H., Bouguettaya A., Motahari H (eds). Service-Oriented Computing. ICSOC 2020. Lecture Notes in Computer Science, vol 12571. Springer, Cham. https://doi.org/10.1007/978-3-030-65310-1_14
13. Li S, Huang J, Cheng B (2021) Resource Pricing and Demand Allocation for Revenue Maximization in IaaS Clouds: A Market-Oriented Approach. IEEE Trans Netw Serv Manag 18(3):3460–3475
14. Chen L, Feng Y, Li B, Li B (2021) A Case for Pricing Bandwidth: Sharing Datacenter Networks With Cost Dominant Fairness. IEEE Trans Parallel Distrib Syst 32(5):1256–1269
15. Applegate D, Archer A, Gopalakrishnan V, Lee S, Ramakrishnan KK (2016) Optimal content placement for a large-scale VoD system. IEEE/ACM Trans Netw 24(6):2114–2127
16. Alizadeh M, Greenberg A, Maltz DA, Padhye J, Patel P, Prabhakar B, Sengupta S, Sridharan M (2010) Data center TCP (DCTCP). ACM SIGCOMM, New Delhi
17. Andrews M (2012) Searching the Internet. IEEE Softw 29(2):13–16
18. Vamana B, Hasan J, Vijakumar T. N (2012) Deadline-Aware Datacenter TCP (D^2TCP). ACM SIGCOMM, Helsinki
19. Roy A, Zeng H, Bagga J, Porter G, Snoeren A. C (2015) Inside the Social Network's (Datacenter) Network. ACM SIGCOMM, London
20. Liu X, Zhang L, Lu K (2020) Image Quality Assessment Method Based on Vector Information and SVD of Quaternion Matrix under Cloud Computing Environments. IEEE Trans Cloud Comput 8(2):326–337. https://doi.org/10.1109/TCC.2015.2513397
21. Dean J, Barroso LD (2013) The Tail at Scale. Commun ACM 56:74–80
22. Wilson C, Ballani H, Karagiannis T, Rowstron A (2011) Better Never than Late: Meeting Deadlines in Datacenter Networks. ACM SIGCOMM
23. Besn T, Akella A, Malta D. A (2010) Network Traffic Characteristics of Data Centers in the Wild. ACM SIGCOMM Conference on Internet Measurement, Melbourne
24. Sun X, Wang Z, Wu Q, Che H, Jiang H (2020) PACCP: A Price-Aware Congestion Control Protocol for Datacenters. In: 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), Leicester. pp 23–33. https://doi.org/10.1109/UCC48980.2020.00022
25. Shenker S (1995) Fundamental Design Issues for the Future Internet. IEEE J Sel Areas Commun 13:1176–1188
26. Wang Z, Singhal A, Wu Y, Zhang C, Che H, Jiang H, Liu B, Lagoa CM (2020) HOLNET: A holistic traffic control framework for datacente networks. IEEE ICNP, Madrid
27. Ye L, Wang Z, Che H, Lagoa CM (2011) TERSE: A Unified End-to-End Traffic Control Mechanism to Enable Elastic, Delay Adaptive, and Rate Adaptive Services. IEEE J Sel Areas Commun 29(5):938–950. https://doi.org/10.1109/JSAC.2011.110504
28. Ye L, Wang Z, Che H, Chan HBC, Lagoa CM, Constantino M (2009) Utility function of TCP, Vol. 32
29. Bertsekas D (1982) Constraint Optimization and Lagrange Multiplier Methods. Computer Science and Applied Mathematics. Academic Press, Boston. https://doi.org/10.1016/B978-0-12-093480-5.50006-4
30. Movsichoff BA, Lagoa C, Che H (2007) End-to-End Optimal Algorithm for Integrated QoS, Traffic Engineering, and Failure Recovery. IEEE Trans Networking 15(4):813–823

31. Kohler E, Handley M, Floyd S (2006) Designing DCCP: congestion control without reliability. ACM SIGCOMM, Pisa
32. Ye L, Wang Z (2009) A QoS-aware congestion control mechanisim for DCCP. In: 2009 IEEE Symposium on Computers and Communications. pp 624–629. https://doi.org/10.1109/ISCC.2009.5202273
33. Narayan A, Cangialosi F, Goyal P, Narayan S, Alizadeh M, Balakrishnan H (2017) The Case for Moving Congestion Control Out of the Datapath. ACM HetNets, Palo Alto
34. Kaufmann A, Stamler T, Peter S, Sharma NK, Krishnamurthy A, Anderson T (2019) TAS: TCP Acceleration as an OS Service. EuroSys, Dresden
35. Jeyakumar V, Alizadeh M, Mazieres D, Prabhakar B, Kim C, Greenberg A (2014) Eyeq: practical network performance isolation at the edge. USENIX NSDI, Berkeley
36. Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S (2011) VL2: A Scalable and Flexible Data Center Network. ACM SIGCOMM Comput Commun Rev 39(4):51–62
37. Niu D, Li B (2013) An efficient distributed algorithm for resource allocation in large-scale coupled systems. IEEE INFOCOM, Turin
38. Wu X, Liu M, Dou W, Gao L, Yu S (2016) A scalable and automatic mechanism for resource allocation in self-organizing cloud. Peer-to-Peer Netw Appl 9(1):28–41
39. Niu D, Feng C, Li B (2012) Pricing cloud bandwidth reservations under demand uncertainty. ACM SIGMETRICS Perform Eval Rev 40(1):151–162
40. Sun X, Zhuo X, Wang Z (2020) A survey of Pricing Aware Traffic Engineering in Cloud Computing. J Internet Technol 21(2):357–364
41. Dimitri N (2020) Pricing cloud IaaS computing services. J Cloud Comput 9(1):1–11
42. Arun V, Hari B (2018) Copa: Practical delay-based congestion control for the internet. ANRW, Montreal
43. Perry J, Balakrishnan H, Shah D (2017) Flowtune: Flowlet control for datacenter networks. USENIX, Santa Clara
44. Ballano H, Costa P, Karagiannis T, Rowstron A (2011) Towards predictable datacenter networks. ACM SIGCOMM, Toronto
45. Guo C, Lu G, Wang HJ, Yang S, Kong C, Sun P, Wu W, Zhang Y (2015) SecondNet: a data center network virtualization architecture with bandwidth guarantees. ACM CoNext, Philadelphia
46. Popa L, Yalagandula P, Banerjee S, Mogul JC, Turner Y, Santos JR (2013) ElasticSwitch: Practical Work-Conserving Bandwidth Guarantees for Cloud Computing. ACM SIGCOMM, Hong Kong
47. Shieh A, Kandula S, Greenberg A, Kim C, Saha B (2011) Sharing the Data Center Network. USENIX NSDI, Boston
48. Lam VT, Radhakrishnan S, Pan R, Vahdat A, Vargese G (2012) Netshare and stochastic netshare: predictable bandwidth allocation for data centers. ACM SIGSOMM Commun Rev 42(3):5–11
49. Lee J, Turner Y, Lee M, Popa L, Banerjee S, Kang J, Sharma P (2014) Application-driven bandwidth guarantees in datacenters. ACM SIGCOMM Comput Commun Rev 44(4):467–478
50. Rodrigues H, Santos JR, Turner Y, Soares P, Guedes D (2011) Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks. USENIX WIOV, Boston
51. Nagaraj K, Bharadia D, Mao H, Chinchali S, Alizadeh M, Katti S (2016) NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters. ACM SIGCOMM, Florianopolis

Publisher's Note