Friends or Foes: Revisiting Strategy-Proofness in Cloud Network Sharing

Wei Wang†, A-Long Jin†‡
†*Hong Kong University of Science and Technology,* ‡*University of Waterloo*
weiwa@cse.ust.hk, along.jin@gmail.com

***Abstract*—Cloud networks consist of a large number of links, on which tenants have *correlated* and *elastic* bandwidth demands in the form of *coflows*. Ideally, a cloud network sharing policy should provide tenants with *isolation guarantees* on the minimum coflow progress, while at the same time attaining as *high utilization* as possible. Prior work shows that to achieve the optimal isolation guarantee, *strategy-proofness* is needed, in that tenants cannot lie about demands to obtain higher progresses. However, this requirement is derived under a simplified assumption that tenants are only interested in maximizing coflow progresses. We show in this work that a rational tenant should pursue more bandwidth allocation as a *secondary objective* after progress maximization. In this new model, enforcing strategy-proofness inevitably hurts the isolation guarantee. We propose a new network sharing policy to achieve the optimal isolation guarantee while attaining the highest possible utilization *in spite of* strategic, untruthful tenants. Trace-driven evaluations show that our policy outperforms existing alternatives with better isolation guarantee, higher utilization, and shorter coflow completion time (CCT).**

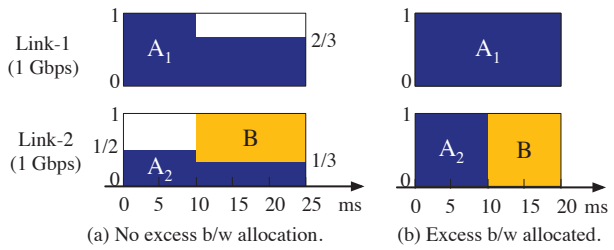
I. Introduction

Achieving optimal isolation guarantees and high utilization constantly shows up as the top requirements in cloud network sharing [1]–[6]. Tenants expect guarantees on the minimum bandwidth to ensure predictable communication performance. Meanwhile, maintaining high utilization allows the cloud network to accommodate more traffics, which may in turn generate higher profits.

Unlike CPU and memory, network demands are *correlated* and *elastic* across datacenter links. Tenants running data-parallel applications such as MapReduce have correlated bandwidth demands on multiple links in the form of *coflows* [7]. For example, consider tenant-A sending 20 Mb and 10 Mb data over two 1 Gbps links, respectively. For every bit it sends on link-1, at least $\frac{1}{2}$ bits should be sent on link-2. Therefore, to achieve the maximal *coflow progress*—defined as the attainable rate on the bottleneck link (link-1 in this example)—we should allocate tenant-A the entire link-1 and *at least* $\frac{1}{2}$ of link-2.

The demands are not only correlated, but also elastic. In the previous example, tenant-A can consume more than half of the capacity of link-2. While this does not lead to a higher coflow progress (1 Gbps), it speeds up the data transfer on that link, improving the overall network utilization.

In this paper, our goal is to achieve the *optimal isolation guarantee* while attaining the *highest utilization* in cloud network sharing. Here, we define the isolation guarantee as the minimum coflow progress a tenant can expect.



(a) No excess b/w allocation. (b) Excess b/w allocated.

Fig. 1: Allocating the spare bandwidth, even though not improving the instantaneous progress, speeds up the coflow completion opportunistically. Tenant-A starts sending 20 Mb and 10 Mb data over two links at 0 ms; tenant-B starts sending 10 Mb data on link-2 at 10 ms. (a) Tenant-A is given the maximal progress without excess bandwidth at time 0. (b) Tenant-A is allocated the full capacity of both links at time 0.

Prior work [3] shows that to achieve the optimal isolation guarantee, *strategy-proofness* must be enforced, in that no tenant can lie about demands to receive a higher progress. To meet this requirement, High Utilization with Guarantees (HUG) [3], has been proposed recently as a promising network sharing policy. HUG's strategy-proofness critically depends on an assumption that a tenant is *only interested* in making a higher coflow progress, yet *indifferent* to receiving more bandwidth without progress improvement.

However, receiving more excess bandwidth, even though not increasing the instantaneous progress, is always beneficial, as it speeds up coflows *opportunistically*. Consider two allocations both giving the maximal coflow progress in the previous example. In Fig. 1a, tenant-A receives no spare bandwidth and is allocated the entire link-1 and $\frac{1}{2}$ of link-2; in Fig. 1b, tenant-A receives the full capacity of both links. One might think that the excess bandwidth allocated on link-2 does not help reduce the coflow completion time (CCT). To show that this is not true, we assume tenant-B starts sending 10 Mb data on link-2 at 10 ms. We see that, without allocating spare bandwidth at the beginning, tenant-A cannot finish transferring data on link-2 by the time tenant-B starts sending, which delays the CCT by 5 ms, or 25%, as compared to that in Fig. 1b.

In general, improving coflow progress reduces CCT *instantaneously*, and is the *primary* objective a tenant should optimize. In addition, coflows can be sped up opportunistically by allocating excess bandwidth. Given that this speedup is

978-1-5090-3281-5/16/$31.00 ©2016 IEEE

indefinite, seeking more bandwidth allocation is a *secondary* objective after progress maximization. That tenants have these *dual objectives* for bandwidth allocation uniquely characterizes the cloud network sharing problem, but it has received little attention before [2], [3], [8], [9]. As we shall show in Sec. III, tenants can game HUG to receive more bandwidth at the same progress. The violation of strategy-proofness questions the optimal isolation guarantee of HUG.

In this paper, we consider tenants seeking more bandwidth allocation second to progress maximization. Contrary to the previous belief [3], we show that strategy-proofness is *no longer necessary* to achieve the optimal isolation guarantee. Rather, there exists a *hard tradeoff* between the two properties: strictly enforcing strategy-proofness inevitably harms the isolation guarantee. On the other hand, allowing strategic manipulations without restrictions can be harmful as well. Ideally, a sharing policy should protect against "harmful" manipulations, while leaving "unharmful" at large.

To meet this requirement, we propose HUG+, a two-stage algorithm that improves HUG with higher utilization. In the first stage, HUG+ seeks a *max-min fair* allocation w.r.t. the progress, i.e., it always maximizes the lowest progress first, followed by the second lowest, etc. In the second stage, HUG+ allocates spare bandwidth such that no tenant is allocated more bandwidth on a link than its progress. This way, HUG+ provides the optimal isolation guarantee while attaining the highest possible network utilization in spite of the strategic, untruthful tenants.

We demonstrate the effectiveness of HUG+ using trace-driven simulations. Compared to HUG, HUG+ improves the progress of 69% of tenants by 2.7× on average without slowing down the others. This translates to 17% higher network utilization. In the long run, HUG+ outperforms HUG by 1.9× in terms of the average coflow completion time. HUG+ scales to large datacenters: even with 10k machines and 1k tenants, a new allocation can be computed in milliseconds on a desktop.

II. Background and Model

In this section, we present our models for cloud network sharing. We highlight two desirable requirements—optimal isolation guarantee and high utilization—that any sharing policy should meet.

A. Background and Model

Thanks to the recent advances in datacenter fabrics [10]–[12], full bisection bandwidth network is now available in production datacenters [13]. This allows us to model the datacenter network as one *non-blocking fabric* where the edges—machine uplinks and downlinks—are the only sources of contention. As shown in Fig. 2, we assume a non-blocking datacenter fabric connecting m machines through full-duplex links, where each ingress (egress) port corresponds to a machine uplink (downlink).

Tenants run data-parallel applications whose communication stage is abstracted out as a *coflow*. A coflow consists of a collection of flows transferring data between groups of machines, e.g., shuffle between map and reduce. In many

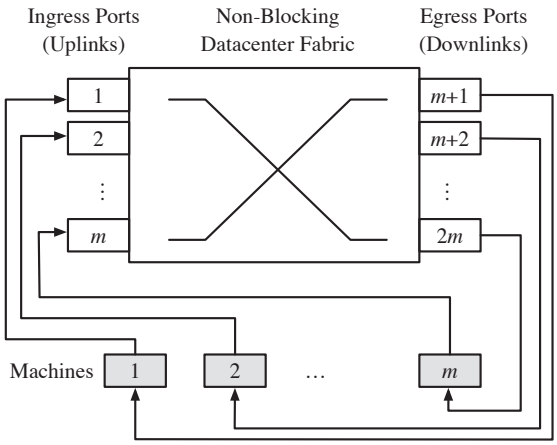


Fig. 2: An $m \times m$ datacenter fabric with m ingress/egress ports connecting to m machines.

data-parallel applications such as MapReduce, the amount of data each flow needs to transfer can be known before the flow starts [3], [14]. We characterize the coflow of each tenant-k by a *correlation vector* $\mathbf{d}_k = \langle d_k^1, \ldots, d_k^{2m} \rangle$, where $d_k^i \leq 1$ and is the normalized bandwidth demand on link-i. Informally, for every bit the coflow transfers on the *bottleneck link* (i.e., those with $d_k^i = 1$), at least d_k^j bits should be sent on link-j.

Given an allocation $\mathbf{a}_k = \langle a_k^1, \ldots, a_k^{2m} \rangle$ where a_k^i is the bandwidth allocated on link-i, the *coflow progress* is defined, for every tenant-k, as the minimum demand-normalized allocation across links:

$$P_k = \min_{1 \leq i \leq 2m} a_k^i / d_k^i. \tag{1}$$

One could interpret the progress as the attainable coflow transmission rate: the higher the progress, the shorter the coflow completion time (CCT). In addition, with elastic demands, allocating more bandwidth—even without any progress improvement—is not wasteful but reduces CCT opportunistically (cf. Fig. 1). We shall show in Sec. VI that allocating spare bandwidth is the main source of CCT reduction.

While high progress and more bandwidth allocation are both desirable to speed up coflows, they play different roles and should be optimized at different priorities. In particular, the CCT reduction given by progress improvement is *guaranteed* and *instantaneous*, regardless of future traffic arrivals. Allocating more bandwidth without progress improvement, on the other hand, reduces the CCT *indefinitely* and *opportunistically*. Therefore, *tenants should always prioritize high progress over more bandwidth allocation*. Specifically, given an allocation, we associate each tenant-k with a utility $U_k = (P_k, A_k)$, where P_k is the achieved progress, and $A_k = \sum_i a_k^i$ is the total amount of bandwidth allocated. The utility is evaluated following a *lexicographic order*, i.e., given two allocations $\mathbf{a}_k$ and $\mathbf{a}'_k$, we have $U_k > U'_k$ *if and only if* $P_k > P'_k$ or $(P_k = P'_k$ and $A_k > A'_k)$.

TABLE I: Summary of notations and definitions.

$\mathbf{d}_k = \langle d_k^1, \ldots, d_k^{2m} \rangle$	Reported correlation vector of tenant-k
$\mathbf{a}_k = \langle a_k^1, \ldots, a_k^{2m} \rangle$	Allocation of tenant-k
$P_k = \min_i a_k^i / d_k^i$	Coflow progress of tenant-k
$A_k = \sum_i a_k^i$	Amount of b/w allocated to tenant-k
$U_k = (P_k, A_k)$	Utility of tenant-k
$\min_k P_k$	Isolation guarantee
$\sum_k A_k$	Network utilization

In this paper, we consider the cloud network sharing problem where tenants compete for bandwidth allocation across links. In particular, we are interested in *non-cooperative* environments like public clouds (e.g., Amazon EC2 [15] and Microsoft Azure [16]), in which each tenant games the reported correlation vector to maximize its utility, i.e., the progress should be maximized first, followed by the total amount of bandwidth allocated. We shall focus on a Nash equilibrium at which *no tenants can unilaterally change its report to improve its utility*.

B. Desirable Sharing Properties

As identified in prior work [1]–[3], there are two desirable properties that should be satisfied by any cloud network sharing policy: optimal isolation guarantee and high utilization.

Optimal isolation guarantee. Tenants expect guarantees on the minimum progress to achieve performance predictability. Specifically, given an allocation, we define the *isolation guarantee* as the minimum progress across tenants, i.e., $\min_k P_k$. A network sharing policy with the *optimal isolation guarantee* gives an allocation that maximizes the minimum progress, i.e.,

$$\text{maximize} \min_k P_k.$$

High utilization. Ideally, bandwidth should not be left in idle if it could be used to serve unsatisfied demands [1], [2]. However, prior work shows that this is *incompatible* with optimal isolation guarantee for cloud network sharing [3]. Given this impossibility result, we require a sharing policy to achieve *as high network utilization as possible without hurting the isolation guarantee*.

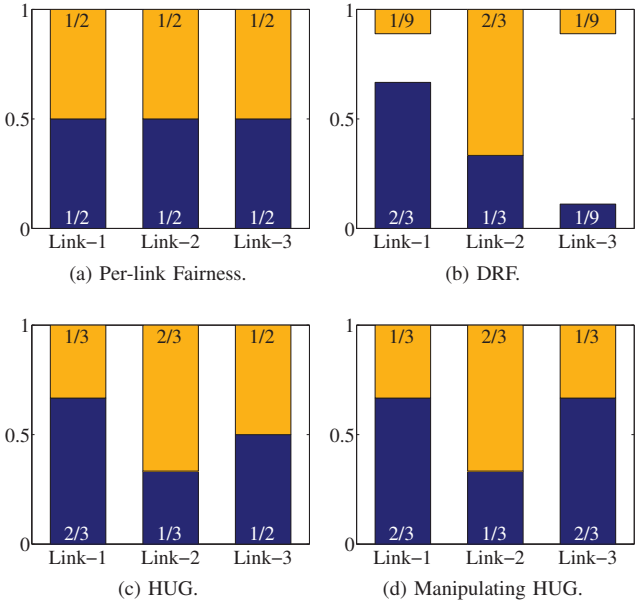
Our goals are to achieve both the optimal isolation guarantee and high utilization. Table I summarizes the important notations and definitions.

III. Inefficiency of Existing Policies

In this section, we briefly survey three popular allocation policies—per-link fairness among tenants [2], [17], DRF [8], [9], and HUG [3]. We show that they either provide suboptimal isolation guarantees or suffer from low utilization.

A. Per-Link Fairness at the Tenant Level

Prior work shows that traditional bandwidth allocation policies, such as fairness among flows, source-destination pairs, or sources alone, provide no isolation guarantees [2]: by initiating more flows or creating more communication endpoints, a tenant can take an arbitrarily high share of network bandwidth, dragging down the progress of others close to 0. To avoid this problem, many recent proposals [2], [17] suggest an alternative allocation that equally divides the capacity of each link among tenants. However, allocations based on per-link fairness among tenants are agnostic to the demand correlation, and are *suboptimal* in isolation guarantees [3].



(a) Per-link Fairness. (b) DRF. (c) HUG. (d) Manipulating HUG.

Fig. 3: An example of two tenants competing on three links, where tenant-A (blue) demands $\mathbf{d}_A = \langle 1, \frac{1}{2}, \frac{1}{6} \rangle$, and tenant-$B$ (orange) demands $\mathbf{d}_B = \langle \frac{1}{6}, 1, \frac{1}{6} \rangle$. (a) Per-link fairness equally divides each link to tenants, and is suboptimal in isolation guarantees. (b) DRF gives the optimal isolation guarantee $\frac{2}{3}$ but results in poor utilization. (c) HUG evenly assigns spare, DRF-unallocated bandwidth to tenants. (d) HUG is not strategy-proof: by claiming $\mathbf{d}'_A = \langle 1, \frac{1}{2}, 1 \rangle$, tenant-$A$ receives more bandwidth on link-3 at the same progress.

Consider an example in Fig. 3, where two tenants, A and B, compete on the uplinks of three machines, on which tenant-A has correlation vector $\mathbf{d}_A = \langle 1, \frac{1}{2}, \frac{1}{6} \rangle$, and tenant-$B$ has $\mathbf{d}_B = \langle \frac{1}{6}, 1, \frac{1}{6} \rangle$. For simplicity, we omit the demands on other non-competing links in the correlation vector. We see from Fig. 3a that the isolation guarantee given by per-link fairness is $\frac{1}{2}$, while the optimum is $\frac{2}{3}$, as shown in Fig. 3b.

B. Dominant Resource Fairness

If we view each link as an independent resource, then the cloud network sharing problem is captured by a multi-resource allocation problem. Optimal isolation guarantees can therefore be achieved by Dominant Resource Fairness (DRF) [8], [9], where each tenant is allocated the same share of bandwidth on the bottleneck link (i.e., dominant resource), and the minimum progress is maximized. Fig. 3b shows the DRF allocation in the previous example. Both tenants receive the same progress $\frac{2}{3}$, which is the maximum as link-2 is already congested.

However, DRF assumes *inelastic* demands and does not allocate spare bandwidth without progress improvement, which

inevitably results in low utilization. Referring back to the example in Fig. 3b, both link-1 and link-3 have unused bandwidth, which accounts for $\frac{1}{3}$ of the total bandwidth availability. Even worse, prior work shows that in some extreme cases, the utilization of DRF could be arbitrarily low [3]. We therefore rule it out as a desirable policy.

C. High Utilization with Guarantees

A simple fix to the utilization problem of DRF is to continue allocating unused bandwidth after the DRF allocation has been done. However, naively allocating unused bandwidth without restrictions allows tenants to game the policy for higher progresses at the expense of others [3, Lemma 1]. It is further shown in [3] that to achieve the optimal isolation guarantee, the policy must be *strategy-proof*, in that the dominant strategy of each tenant is to *truthfully* report the correlation vector.

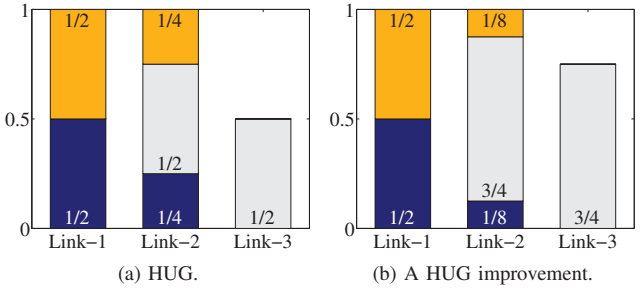
High Utilization with Guarantees (HUG) [3] has been proposed as a two-stage algorithm to meet the strategy-proofness requirement. In the first stage, HUG achieves the optimal isolation guarantee by equally increasing the progress of every tenant to the maximum level, computed as

$$P^* = \frac{1}{\max_i \sum_k d_k^i}. \tag{2}$$

In the second stage, HUG maximizes the utilization by evenly allocating unused bandwidth among tenants in a "strategy-proof" manner. Specifically, for every tenant-k, the amount of bandwidth it receives on a link should not exceed its progress, i.e., $a_k^i \leq P^*$ for all link-i. Continuing the previous example in Fig. 3, HUG gives the same allocation as DRF in the first stage (Fig. 3b). In the second stage, spare bandwidth is evenly allocated to the two tenants. Because tenant-A has already reached its allocation cap $\frac{2}{3}$ on link-1, all the spare bandwidth goes to tenant-B (Fig. 3c). Compared to DRF, HUG achieves 100% network utilization in this example.

Two concerns. HUG offers a promising solution to sharing the cloud network. However, there are two concerns that have not been noted before. First, by capping the allocation of spare bandwidth, HUG ensures that tenants cannot achieve higher progresses by gaming the policy, but it is still possible for them to lie about demands to receive more bandwidth at the same progress. Back to the previous example in Fig. 3, suppose tenant-A lies and misreports a different correlation vector $\langle 1, \frac{1}{2}, 1\rangle$, while tenant-$B$ remains truthful. Fig. 3d shows the allocation given by HUG. We see that by claiming more demands on link-3, tenant-A successfully increases its allocation on that link to $\frac{2}{3}$ at the same progress. Therefore, HUG is *not strategy-proof*, which makes its optimal isolation guarantee a questionable claim.

In addition, HUG enforces the *same* progress across tenants (i.e., P^* given by (2)), which may result in low utilization that cannot be made up by allocating spare bandwidth in the second stage. Consider an example of three tenants shown in Fig. 4, where $\mathbf{d}_A = \mathbf{d}_B = \langle 1, \frac{1}{4}, 0\rangle$ and $\mathbf{d}_C = \langle 0, 1, \frac{1}{2}\rangle$. Fig. 4a shows the corresponding HUG allocation, where the progress of all tenants is equally increased to $\frac{1}{2}$ in the first stage, followed by



(a) HUG. (b) A HUG improvement.

Fig. 4: An example of three tenants competing on three links, where both tenant-A (blue) and tenant-B (orange) have $\mathbf{d}_A = \mathbf{d}_B = \langle 1, \frac{1}{4}, 0\rangle$, and tenant-$C$ (gray) demands $\mathbf{d}_C = \langle 0, 1, \frac{1}{2}\rangle$. (a) HUG enforces the same progress $\frac{1}{2}$ across tenants, leaving half of link-3 idle. (b) Increasing the progress of tenant-C leads to higher utilization without compromising isolation guarantee.

the allocation of spare bandwidth in the second stage. Because tenant-C cannot receive more bandwidth beyond its progress $\frac{1}{2}$, half of link-3 is left unallocated. This is wasteful, as tenant-C could have a higher progress and more allocation quota for spare bandwidth. Consider a HUG improvement in Fig. 4b, where tenant-C's progress is increased to $\frac{3}{4}$. We see that the isolation guarantee remains $\frac{1}{2}$—the same as HUG—but the utilization is higher.

We shall show in Sec. VI that the inefficiency of HUG illustrated above is not a rare case but generally found when replaying the production traces. Unless *all* tenants have demands on *all* machine links—which is practically impossible given the huge number of machines in a datacenter—enforcing the same coflow progress likely wastes bandwidth, the degree of which depends on the number of tenants. In fact, we show in the following theorem that the utilization of HUG could be *arbitrarily low*. The proof is deferred to our technical report [18] due to the page limit.

Theorem 1 (Inefficiency of HUG): The utilization of HUG can be arbitrarily close to 0.

To summarize, existing allocation policies are either suboptimal in isolation guarantees or unable to achieve high utilization. It remains unclear how the cloud network should be shared among tenants with correlated and elastic demands.

IV. Strategy-Proofness—Foes, not Friends

Prior work [3] identifies strategy-proofness as a *necessary condition* to achieve the optimal isolation guarantee. We stress that this result is obtained under a simplified, yet impractical assumption that tenants are only concerned with the instantaneous progress (i.e., $U_k = P_k$ for all tenant-k). However, we see from the previous discussions that a *fully rational* tenant should seek more bandwidth allocation once its progress has reached the maximum (i.e., $U_k = (P_k, A_k)$ for all tenant-k). Contrary to the previous belief, we show through the following theorem that enforcing strategy-proofness among fully rational tenants instead harms the isolation guarantee.

Theorem 2 (Impossibility): No strategy-proof policy can achieve an isolation guarantee at level

$$P^* = \frac{1}{\max_i \sum_k d_k^i}. \quad (3)$$

Proof: Let us suppose the opposite. In particular, we assume that there exists a strategy-proof policy $\mathcal{P}$ that can provide an isolation guarantee at level P^*, i.e., $P_k \geq P^*$ for all tenant-k. Consider an example of three tenants, A, B, and C, competing on three links, where $\mathbf{d}_A = \mathbf{d}_B = \langle 1, \epsilon, 0\rangle$ and $\mathbf{d}_C = \langle 0, \epsilon, 1\rangle$. Here, we let $\epsilon \to 0$. By assumption, policy $\mathcal{P}$ ensures that the progress of every tenant is at least $P^* = \frac{1}{2}$. Let the allocation given to tenant-A and B be respectively denoted as $\mathbf{a}_A = \langle \frac{1}{2}, x_A, 0\rangle$ and $\mathbf{a}_B = \langle \frac{1}{2}, x_B, 0\rangle$. We have

$$\frac{\epsilon}{2} \leq x_A, x_B \leq \frac{1}{2},$$

where the lower bound $\frac{\epsilon}{2}$ holds because $P_A, P_B \geq \frac{1}{2}$, and the upper bound $\frac{1}{2}$ is required to ensure strategy-proofness (cf. [3, Lemma 1] or Theorem 3 presented later). Because tenant-C has demand on link-2, the total bandwidth allocated to tenant-A and B on that link is less than 1 (i.e., $x_A + x_B < 1$), which suggests that x_A and x_B cannot be both $\frac{1}{2}$. Without loss of generality, we assume $x_A < \frac{1}{2}$.

Now assume that tenant-A claims more demand on link-2 with $\mathbf{d}'_A = \langle 1, 1, 0\rangle$, while the other two tenants remain truthful. The target isolation guarantee remains the same, i.e., $P'^* = \frac{1}{2}$. Because there is no way for policy $\mathcal{P}$ to tell if tenant-A is lying or not, to achieve the target isolation guarantee, the policy must allocate tenant-A at least $\mathbf{a}'_A = \langle \frac{1}{2}, \frac{1}{2}, 0\rangle$. As a result, tenant-$A$ receives more bandwidth on link-2 at the same progress by lying, which contradicts our previous assumption that policy $\mathcal{P}$ is strategy-proof. ■

Because Theorem 2 is stated for strategy-proof policies, we do not differentiate between the reported demands and the true demands in the calculation of P^*. We shall show in Sec. V-B (Theorem 4) that the isolation guarantee of P^* is achievable, which immediately leads to the following corollary:

Corollary 1 (Hard tradeoff): No strategy-proof policy is optimal in isolation guarantees.

In other words, Corollary 1 indicates that strategy-proofness is a "foe," not a "friend" of isolation guarantee. For example, per-link fairness among tenants (Sec. III-A) is strategy-proof, but it is suboptimal in isolation guarantees (cf. Fig. 3a).

Given the hard tradeoff between strategy-proofness and isolation guarantee, we can (1) either retain the former first, then optimize the later; or (2) optimize isolation guarantee in the presence of strategic, untruthful tenants. A general consensus summarized in [1] reveals that tenants care more about isolation guarantee than allocation "fairness." We therefore do not give up on the optimality of isolation guarantee, but are tolerant of strategic behaviors.

We caution that Corollary 1 should not be misinterpreted as if strategic manipulations were "helpful" and should be allowed in all cases. In fact, it is shown in [3, Lemma 1] that some manipulations are *harmful* to the isolation guarantee, which is rephrased in the following theorem.

Theorem 3 ([3, Lemma 1]): Any allocation policy meeting the following two conditions results in strategic manipulations that can hurt the isolation guarantee:

1) it first uses DRF to increase the progress of every tenant to P^* given by (2), and then allocates spare bandwidth;
2) there exists a tenant-k whose allocation on a link-i is more than its progress, i.e., $a_k^i > P_k$.

Intuitively, we need to differentiate between two types of strategic manipulations, harmful to the isolation guarantee and unharmful. To achieve the optimal isolation guarantee, *an allocation policy should guard against harmful manipulations, while leaving unharmful ones at large*. We next show how this can be achieved by a HUG improvement algorithm.

V. HUG+: Design and Analysis

In this section, we present HUG+, a HUG improvement algorithm that provides the optimal isolation guarantee while achieving the highest attainable network utilization.

A. HUG+

We see from Sec. III-C that the root cause behind the inefficiency of HUG is the enforcement of the same progress across tenants: a tenant that could have a higher progress (e.g., tenant-C in the example in Fig. 4) is forced to keep it as low as others. Because HUG does not allocate more bandwidth on a link beyond a tenant's progress, having as high progress as possible for every tenant in the first place is the key to achieving high utilization.

Following this intuition, we propose HUG+, a two-stage algorithm that improves HUG with higher utilization. In the first stage, HUG+ seeks a *max-min fair* allocation w.r.t. progress. Specifically, the algorithm starts by equally increasing the progress of all tenants to the maximum in round-1. It then decides which tenant's progress can be further increased without lowering that of others. All these tenants remain *active*, who are those having no demand on congested links (no spare bandwidth). The other tenants become *inactive* and have their allocations *frozen*. The algorithm proceeds to round-2, in which the progress of all active tenants is equally increased to a new maximum level. The entire process repeats until there are no more active tenants. Algorithm 1 gives the details, where the computation of Stage-1 resembles *multi-round* DRF [9] applied to $2m$ resources.

In the second stage, HUG+ allocates unused bandwidth to increase the network utilization. By Theorem 3, allocating spare bandwidth without restrictions hurts the isolation guarantee. Therefore, we restrict the allocation such that no tenant is assigned more bandwidth on a link than its progress, i.e.,

$$a_k^i \leq P_k, \quad \text{for all tenant-}k \text{ and link-}i.$$

Example. We use the example in Fig. 4 to illustrate how HUG+ works. In the first stage, HUG+ starts by equally increasing the progress of all three tenants to $\frac{1}{2}$, which is the maximum as link-1 is congested. Because both tenant-A

Algorithm 1 HUG+

Input: Correlation vector $\{\mathbf{d}_k\}$ reported by every tenant-k
Output: Bandwidth allocation $\{\mathbf{a}_k\}$ given to every tenant-k
Stage 1: Compute a max-min fair allocation w.r.t. progress using multi-round DRF [9]
$r \leftarrow 1$ ▷ Current round
$b_i \leftarrow 1,\ 1 \le i \le 2m$ ▷ Available b/w on link-i
$\mathcal{A}_r \leftarrow \{1, \ldots, n\}$ ▷ Active tenants in round r
$\mathbf{a}_k \leftarrow \mathbf{0},\ k \in \mathcal{A}_r$ ▷ Allocation of tenant-k
$P_k \leftarrow 0,\ k \in \mathcal{A}_r$ ▷ Progress of tenant-k
while $\mathcal{A}_r \neq \emptyset$ **do**
Compute the maximum progress increment in round r:

$$\Delta_r \leftarrow \min_{1 \le i \le 2m} \frac{b_i}{\sum_{k \in \mathcal{A}_r} d_k^i} \tag{4}$$

for all active tenant-k, $k \in \mathcal{A}_r$ **do**
$P_k \leftarrow P_k + \Delta_r$ ▷ Update tenant-k's progress
$\mathbf{a}_k \leftarrow \mathbf{a}_k + \Delta_r \mathbf{d}_k$ ▷ Allocate b/w to tenant-k
$b_i \leftarrow 1 - \sum_k a_k^i$ ▷ Update available b/w on all link-i
$r \leftarrow r + 1$ ▷ Proceed to the next round
$\mathcal{A}_r \leftarrow \{k : \forall i, d_k^i > 0 \Rightarrow b_i > 0\}$ ▷ Tenants having no demand on congested links remain active
Stage 2: Restrict the allocation of spare bandwidth to tenants on all $2m$ links, such that $a_k^i \le P_k$ for all tenant-k and link-i

TABLE II: Illustration of HUG+ allocation in the example of Fig. 4, where $\mathbf{d}_A = \mathbf{d}_B = \langle 1, \frac{1}{4}, 0\rangle$ and $\mathbf{d}_C = \langle 0, 1, \frac{1}{2}\rangle$.

Stage-Round	Tenant-A	Tenant-B	Tenant-C
S1-R1	$\langle \mathbf{\frac{1}{2}}, \mathbf{\frac{1}{8}}, 0\rangle$	$\langle \mathbf{\frac{1}{2}}, \mathbf{\frac{1}{8}}, 0\rangle$	$\langle 0, \mathbf{\frac{1}{2}}, \mathbf{\frac{1}{4}}\rangle$
S1-R2	$\langle \frac{1}{2}, \frac{1}{8}, 0\rangle$	$\langle \frac{1}{2}, \frac{1}{8}, 0\rangle$	$\langle 0, \mathbf{\frac{3}{4}}, \mathbf{\frac{3}{8}}\rangle$
S2	$\langle \frac{1}{2}, \frac{1}{8}, 0\rangle$	$\langle \frac{1}{2}, \frac{1}{8}, 0\rangle$	$\langle 0, \frac{3}{4}, \mathbf{\frac{3}{4}}\rangle$

and tenant-B have demands on link-1, their progresses cannot be further increased. The two tenants become inactive, and their allocations are frozen. The algorithm proceeds to round-2, where tenant-C is the only one active. HUG+ increases its progress to $\frac{3}{4}$, at which link-2 is congested. Tenant-C then becomes inactive. Because there are no more active tenants, Stage-1 concludes, and Stage-2 kicks in. Tenant-C is allocated the spare bandwidth on link-3, increasing its allocation to its progress $\frac{3}{4}$. The algorithm halts and ends up with the same allocation illustrated in Fig. 4b. Table II summarizes the entire allocation process.

B. HUG+ Properties

While HUG+ is a simple improvement over HUG, we show that it provides the optimal isolation guarantee while attaining the highest possible utilization.

Optimal isolation guarantee. We have shown in Sec. IV that to achieve the optimal isolation guarantee, strategy-proofness should not be enforced. This is indeed the case in HUG+. Referring back to the previous example in Fig. 3, we see that HUG+ ends up with the same allocation as HUG (Fig. 3c), and a tenant (i.e., tenant-A in Fig. 3d) can lie about its correlation vector to receive more bandwidth at the same progress. Nevertheless, the isolation guarantee remains at level $\frac{2}{3}$, the same as before (Fig. 3c vs. Fig. 3d). We next show that this is not a coincidence.

Because tenants can lie about demands, we need to differentiate between the *true progress* and *revealed progress*: the former is computed based on the *true correlation vector* of a tenant, and the latter based on the reported correlation vector. Formally, let $\bar{\mathbf{d}}_k$ and $\mathbf{d}_k$ respectively denote the true and reported correlation vector of tenant-k. Given an allocation $\mathbf{a}_k$, the true progress of tenant-k is

$$\bar{P}_k = \min_{1 \le i \le 2m} a_k^i / \bar{d}_k^i, \tag{5}$$

while the revealed progress is $P_k = \min_i a_k^i / d_k^i$. An allocation policy should provide isolation guarantees on the true progress. However, the challenge is: without strategy-proofness, we cannot tell if the revealed progress is true or not, let alone the optimality of isolation guarantees.

To address this challenge, we consider a Nash equilibrium at which no tenant can unilaterally change its report to improve the utility. We show through the following theorem that such an equilibrium not only exists, but also provides guarantees on the minimum true progress. The proof is deferred to our technical report [18] due to the space constraints.

Theorem 4 (Isolation guarantee): With HUG+, there exists a Nash equilibrium at which the true progress of each tenant-k is at least P^*, i.e.,

$$\bar{P}_k \ge P^* = \frac{1}{\max_i \sum_l \bar{d}_l^i}. \tag{6}$$

Recall that we have shown in Theorem 2 that the isolation guarantee at level P^* is not achievable by a strategy-proof policy.[1] HUG+ breaks this glass ceiling by selectively allowing manipulations that are "unharmful" to the isolation guarantee.

It is straightforward to show that the isolation guarantee P^* offered by HUG+ in Theorem 4 is the optimum tenants can expect. We hence omit the proof of the following theorem.

Theorem 5 (Optimal isolation guarantee): HUG+ provides the optimal isolation guarantee.

High utilization. We next show that HUG+ achieves the highest attainable utilization.

Theorem 6 (High utilization): Among all policies with optimal isolation guarantees, HUG+ attains the highest utilization.

Proof: Consider an allocation policy $\mathcal{P}$ with the optimal isolation guarantee. By Theorem 4 and 5, policy $\mathcal{P}$ provides the isolation guarantee at level P^*. By Theorem 3, policy $\mathcal{P}$ allocates a tenant no more bandwidth on a link than its progress. Policy $\mathcal{P}$ therefore achieves no higher utilization than HUG+, as the latter allocates each tenant exactly the same amount of bandwidth on a link as its progress, if possible. ■

In addition to the optimal isolation guarantee and high utilization, HUG+ trivially retains *min-cut proportionality* [3], a desirable property possessed by HUG. In a nutshell, this property ensures that each tenant receives the minimum bandwidth in proportion to the size of the *minimum cut* [19]

[1] In Theorem 2, there is no need to differentiate between the true demand $\bar{\mathbf{d}}_k$ and the reported demand $\mathbf{d}_k$.

TABLE III: Properties of per-link fairness, multi-round DRF, HUG, and HUG+: strategy-proofness (SP), optimal isolation guarantee (IG), and high utilization (HU), where SP and optimal IG are incompatible with each other.

	SP	Optimal IG	HU
Per-link Fairness [2]	√	×	×
Multi-round DRF [9]	×	√	×
HUG [3]	×	√	×
HUG+	×	√	√

of its coflow communication pattern (e.g., many-to-one, many-to-many, etc.). We refer to [3] for detailed discussions.

To summarize, we compare HUG+ in Table III against the other three allocation policies described in Sec. III. We see that HUG+ is the *only* policy that obtains the maximal sets of non-conflicting properties.

C. Practical Consideration

Like HUG [3], HUG+ can be easily incorporated into the prevalent cloud monitoring services like Amazon CloudWatch [20]. Through public APIs, tenants periodically communicate correlation vectors with a central controller, who computes new allocations and notifies the monitoring agent running on each machine for local enforcement. The controller only needs to compute the progress of each tenant and updates local agents. Upon receiving the progress update, the agent running on machine-i respectively enforces allocation $P_k d_k^i$ and $P_k d_k^{m+i}$ on the uplink and downlink for tenant-k. It then allocates spare bandwidth evenly to local tenants following Stage-2 of Algorithm 1. Compared to HUG, the implementation overhead mainly comes from the computation of multi-level progresses in rounds in the central controller. We show in the next section that this overhead is not a scalability concern.

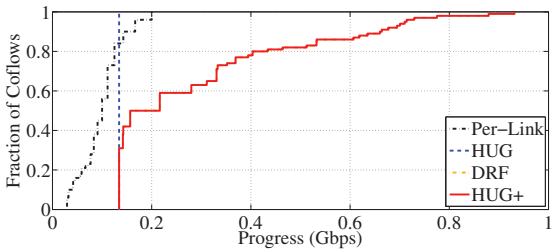
VI. Evaluation

We evaluated HUG+ against various allocation policies—per-link fairness (e.g., PS-P [2]), HUG [3], and multi-round DRF [9]—using trace-driven simulations. We start by characterizing the performance of instantaneous allocations w.r.t. coflow progress, isolation guarantees, and utilization. We then evaluate how optimizing instantaneous allocations could reduce the coflow completion time in the long run. Finally, we study the scalability of HUG+ by comparing its computational overhead against HUG in a simulated large-scale datacenter. The highlights of our evaluation are summarized as follows:

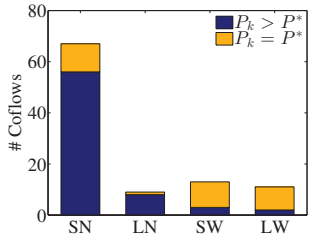
- Regarding the instantaneous allocations, HUG+ provides the maximal isolation guarantee at the highest utilization. Compared to HUG, HUG+ speeds up the progress of 69% of tenants by 2.7× on average without slowing down the others, leading to 17% higher utilization.
- HUG+ outperforms all alternative policies in the long run, reducing the average CCT by 1.9× and 1.5× as compared with HUG and multi-round DRF, respectively.
- HUG+ scales to large datacenters and can be computed in milliseconds in a simulated 10,000-machine cluster. The computational overhead is less than 4× of that of HUG.

TABLE IV: Coflows binned by their lengths (**S**hort or **L**ong) and widths (**N**arrow or **W**ide) in Coflow-Benchmark [21].

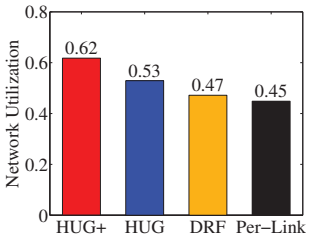
Bin	SN	LN	SW	LW
% of Coflows	60%	16%	12%	12%



(a) CDF of Coflow Progress.



(b) Improved coflows under HUG+.



(c) Network utilization.

Fig. 5: Characteristics of instantaneous allocation with 100 concurrent coflows using different policies.

A. Workload Description and Simulation Setup

Workload. We use the one-hour workload trace in Coflow-Benchmark [21] as the input of our simulation. The workload is synthesized based on a Hive/MapReduce trace collected from a 3000-machine cluster with 150 racks at Facebook [14]. The trace contains 526 coflows scaled down to a 150-port fabric, where all mappers (reducers) in the same rack are combined into one rack-level mapper (reducer). For each coflow, the trace logs its arrival time, placements of mappers/reducers, and the amount of data shuffled, based on which we can easily calculate the correlation vector using the optimal rate allocation algorithm given in [14], [22].

We categorize coflows into four bins based on their lengths and widths. Following the convention [3], [7], [14], [23], we say a coflow is *short* (*long*) if the size of its longest flow is less (greater) than 5 MB, and *narrow* (*wide*) if the number of its flows is less (greater) than 50. Table IV details the distribution of coflows in four bins.

Simulation setup. In our simulation, we assume a non-blocking 150×150 datacenter fabric with 150 ingress/engress ports corresponding to the uplinks/downlinks of 150 racks connected to it. The bandwidth capacity of each port is set to 1 Gbps. We assume that each coflow corresponds to a tenant. We implemented all the allocation policies—HUG+, HUG [3], per-link fairness [2], and multi-round DRF [9]—atop CoflowSim [24], the de facto simulator for coflow scheduling.

B. Instantaneous Allocation Performance

We start our evaluation by comparing instantaneous allocation performance under different policies. In particular, we took a snapshot of 100 coflows randomly sampled from the trace [21]. For each coflow, we computed its allocation and progress using different policies. We have repeated the simulations several times feeding different random samples of coflows, and have observed consistent performance. We report the result of one simulation run as follows.

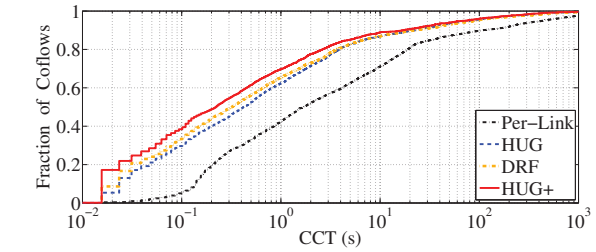
Coflow progress. Fig. 5a shows the distribution of coflow progress under per-link fairness, HUG, multi-round DRF, and HUG+. By design, the latter two policies give the same progress to each tenant, and their CDF curves align with each other. We see that per-link fairness performs the worst in terms of isolation guarantee—only 22% of that of the other three policies—and has the lowest mean progress (99 Mbps). We attribute the poor guarantee to per-link fairness's being agnostic to the demand correlation across links. This problem is avoided using the other three policies, all giving the optimal guarantee on the minimum progress. However, unlike HUG+ (multi-round DRF), HUG refrains tenants from a higher progress beyond the minimum guarantee, which slows down 69% of coflows by 2.7× on average. As a result, the mean progress of HUG is only 46% of that using HUG+ (multi-round DRF).

Given the salient progress improvement of HUG+, we are curious about which coflows have been benefited with higher progresses than the minimum guarantee, i.e., $P_k > P^*$. Fig. 5b shows the number of these coflows in four bins. We see that among all 69 coflows with a higher progress beyond the isolation guarantee, 64 are narrow. Intuitively, compared with wide ones, narrow coflows have demands on fewer links, and are less likely bottlenecked on the congested links when the minimum progress guarantee is reached in Stage-1 of HUG+, allowing them to have a higher progress than the guarantee.

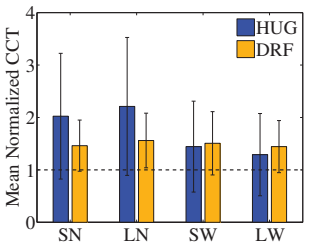
Network utilization. Fig. 5c shows the instantaneous utilization using different policies. We see that HUG+ achieves the highest utilization among all four alternatives, and is 17% more efficient than the second best (HUG). In addition to this expected result, we make another two observations that seem counterintuitive at the first glimpse.

First, we see that per-link fairness performs the worst. This is somehow unexpected, as per-link fairness is *work conserving* [2], and will not leave bandwidth unallocated in the presence of unmet demands. We found that the root cause behind is that the bandwidth, though allocated, cannot be utilized due to the *flow conservation* constraint [19], i.e., the total amount of ingress traffics must match that of egress traffics. Therefore, more bandwidth allocations, either for ingress or egress traffics, are wasteful. This has been commonly observed in per-link fairness for its being agnostic to the demand correlation.

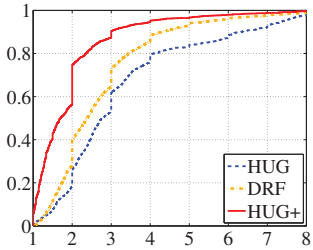
Second, given the significant progress improvement of multi-round DRF over HUG (Fig. 5a), we might expect that the former outperforms the latter in utilization as well. However, an opposite observation is drawn in Fig. 5c. This is due to the fact that DRF does not allocate excess bandwidth without progress improvement. In our experiment, the utilization loss due to unallocated spare bandwidth outweighs the utilization gains derived from higher progress.



(a) CDF of coflow completion time (CCT).



(b) Normalized CCT in bins.



(c) CDF of Slowdown.

Fig. 6: Long-term characteristics using different policies.

C. Long-Term Performance

We have shown in the previous evaluations that HUG+ outperforms all the other policies with the optimal instantaneous performance, such as the isolation guarantee and utilization. We next evaluate how the instantaneous optimality translates to long-term performance gains in terms of the CCT reduction. In particular, to simulate a congested environment with more competitions among tenants, we overloaded the datacenter fabric by linearly scaling up the one-hour trace of 526 coflows [21] to 2630 coflows arriving in two hours.

Fig. 6a shows the distribution of CCT using different allocation policies. We see that per-link fairness results in significantly longer CCT than that of the other three policies. This observation is consistent with that drawn in [3]. Given the clear disadvantage of per-link fairness, we exclude it from further comparison. In addition, we see from Fig. 6a that HUG+ achieves the shortest CCT, but the speedup seems not salient enough to be visually identifiable. We stress that this indistinguishability is purely due to the visualization of using log-scale x-axis in Fig. 6a. To better illustrate the difference between HUG+ and the other two policies, we instead use the following two metrics for comparison: normalized CCT and shuffle slowdown [3].

- *Normalized CCT* is defined, for each coflow, as its duration using the compared policy normalized by that using HUG+, i.e.,

$$\text{Normalized CCT} = \frac{\text{Coflow Duration}}{\text{Duration using HUG+}}.$$

TABLE V: Statistical summary of slowdown.

	Per-Link Fairness	HUG	DRF	HUG+
Min	1.00	1.00	1.00	1.00
Mean	15.44	3.28	2.74	1.96
95th	30.56	7.40	5.50	4.00
Std. Dev.	7.71	1.78	1.37	1.13

- *Shuffle slowdown* is defined, for each coflow, as its duration using the compared policy normalized by the minimum possible duration if it were the only coflow sending in the datacenter fabric, i.e.,

$$\text{Slowdown} = \frac{\text{Coflow Duration}}{\text{Minimum Duration}}.$$

Normalized CCT. We present the mean normalized CCT of HUG and multi-round DRF in four coflow bins in Fig. 6b, where the error bar captures one standard deviation. We observed the average normalized CCT consistently greater than 1 across all bins—meaning that HUG+ speeds up coflows *indistinguishably*. In particular, HUG+ reduces the average CCT of HUG and multi-round DRF by 1.9× and 1.5×, respectively.

Slowdown. A clear win for HUG+ is also observed in Fig. 6c, where the distributions of slowdown under the three policies are given for comparison. We summarize in Table V the min, average, and 95th percentile slowdown as well as the standard deviation measured under different policies. HUG+ consistently outperforms the other three alternatives, reducing the 95th percentile slowdown of HUG and multi-round DRF by 1.9× and 1.4×, respectively.

HUG vs. multi-round DRF. Both normalized CCT (Fig. 6b) and slowdown (Fig. 6c) indicate that coflows have experienced longer completion time under HUG than under multi-round DRF. Recall that HUG is characterized by high utilization but low progress, and multi-round DRF the other way around (cf. Sec. VI-B). The fact that HUG results in longer CCT than DRF validates our tenant model that receiving more bandwidth is less beneficial than having a higher progress, and is the secondary objective tenants set to optimize (cf. Sec. II). HUG+ improves both the progress and the utilization of spare bandwidth, achieving the best of both worlds.

D. Scalability

Prior work shows that, for a centralized resource allocator like HUG to scale, the computation of a new allocation should be done in milliseconds [3]. Compared to HUG, the main overhead introduced by HUG+ is the computation of multi-level progresses in rounds (i.e., Stage-1 of Algorithm 1). To evaluate the scalability of HUG+, we compared the computational overhead of HUG+ against HUG in a large, simulated cluster.

Specifically, we simulated a 10,000-machine datacenter with 1,000 randomly generated coflows competing for bandwidth, using both HUG and HUG+. We run the simulation 1,000 times on a desktop with 2.6 GHz Intel® Core™ i7-5600U processor and 8 GB memory. HUG took about 16.3 milliseconds on average to compute a new allocation, while HUG+ took 52.8 milliseconds on average, less than 4× of that required by HUG. Based on the measurements given in [3], the additional computational overhead has little impact on the scalability.

VII. Related Work

Fairness at flow/machine level. Traditional network sharing mechanisms rely on per-flow fairness [25], [26] to provide isolation guarantee. However, per-flow fairness is susceptible to manipulation: tenants can obtain an arbitrarily high share of network bandwidth by initiating more flows. To avoid this problem, recent work focuses on machine-level fairness. Notably, Seawall [4] suggests per-source fair sharing on the congested links. However, per-source (per-destination) allocation can be unfair to destinations (sources). Popa et al. [2] further shows that per source-destination pair allocation is not fair for tenants either. The lack of isolation guarantees of flow- or machine-level allocation policies results in highly volatile, unpredictable services [1], [2]. In contrast, HUG+ provides tenant-level isolation guarantees with predictable network performance.

Tenant-level reservation. Static, reservation-based bandwidth allocation policies have been implemented in recent systems to achieve performance predictability. For example, SecondNet [6] provides tenants with bandwidth guarantees through the abstraction of virtual datacenter. Oktopus [5] implements the abstraction of virtual network, which isolates tenant performance from the underlying infrastructure and allows tenants to express their network requirements. Pulsar [27] provides guarantees on the end-to-end throughput through the abstraction of a virtual datacenter dedicated to each tenant. All these systems provide network performance isolation atop flexible virtual topologies. However, the enforcement of static reservation does not allow tenants to use the spare bandwidth of another, making the network utilization a major concern of these systems. HUG+ avoids this problem by dynamically allocating spare bandwidth to tenants.

Network-wide/Tenant-level fair sharing. In cooperative environments where strategy-proofness is not a concern, work conservation is required to achieve the highest utilization. Systems like NetShare [28] are work conserving, but provide no isolation guarantee. PS-P [2] and EyeQ [17] address this problem with bandwidth guarantees, but at a suboptimal level. In addition, their implementations require expensive hardware support in switches. HUG [3] comes as an attractive solution with the optimal isolation guarantee. HUG is work conserving as well: since strategy-proofness is not a concern, spare bandwidth can be allocated without restrictions. HUG+ retains all these desirable properties, and further allows tenants to receive higher progresses beyond the minimum guarantee.

In non-cooperative environments like public clouds, work conservation is no longer desirable as it could be at odds with the optimal isolation guarantee. Instead, strategy-proofness is believed necessary [3], if tenants are only interested in making a higher progress. HUG serves as an ideal solution in this model, where it achieves the optimal isolation guarantee, while attaining high utilization without compromising strategy-proofness [3]. However, we have shown in this paper that a

rational tenant should always pursue more bandwidth after its progress reaches the maximum. In this case, strategy-proofness cannot coexist with the optimal isolation guarantee. HUG+ retains the maximal sets of non-conflicting properties, replacing HUG as a desirable network sharing policy.

Multi-resource allocation. Cloud network sharing is in essence a multi-resource allocation problem, if we view each link as a resource. Existing multi-resource allocation policies, such as DRF [8], [9] and its variants [29], [30], assume *inelastic* demands where user utilities are captured by Leontief preferences [30], [31]. Ignoring demand elasticity in cloud network sharing inevitably results in low utilization [3].

VIII. Concluding Remark

In this paper, we have studied the cloud network sharing problem for tenants with correlated and elastic demands on multiple links. Our goals are to achieve both the optimal isolation guarantee and high utilization. Contrary to the previous belief, we have shown that to achieve the optimal isolation guarantee, strategy-proofness should not be enforced. On the other hand, allowing strategic manipulations without restrictions could do more harm than good.

We have addressed this "dilemma" with HUG+, a two-stage bandwidth allocation algorithm. In the first stage, HUG+ maximizes the isolation guarantee using multi-round DRF; it then allocates spare bandwidth such that no tenant is assigned more bandwidth than its progress. We have shown that HUG+ provides the optimal isolation guarantee while attaining the highest possible network utilization in spite of strategic, untruthful tenants. Experimental results show that by achieving these properties, HUG+ outperforms existing network sharing policies with better isolation guarantee, higher utilization, and shorter coflow completion time.

We have focused on evaluating HUG+ using workload traces collected from production clusters. In the future, theoretical aspects should be investigated, such as the equilibrium of strategic behaviors and the Price of Anarchy [32, Ch. 17]. Furthermore, the evaluation of other system performance, such as the operator's revenue derived from the tenants' payments for network usage, is interesting and relevant in public clouds.

References

[1] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 44–48, 2012.

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

[3] M. Chowdhury, Z. Liu, A. Ghodsi, and I. Stoica, "HUG: Multi-resource fairness for correlated and elastic demands," in *USENIX NSDI*, 2016.

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

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

[6] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: A data center network virtualization architecture with bandwidth guarantees," in *ACM CoNEXT*, 2010.

[7] M. Chowdhury, "Coflow: A networking abstraction for distributed data-parallel applications," Ph.D. dissertation, University of California, Berkeley, 2015.

[8] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *USENIX NSDI*, 2011.

[9] D. C. Parkes, A. D. Procaccia, and N. Shah, "Beyond dominant resource fairness: extensions, limitations, and indivisibilities," *ACM Transactions on Economics and Computation*, vol. 3, no. 1, p. 3, 2015.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *ACM SIGCOMM*, 2009.

[11] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM*, 2009.

[12] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, and G. Varghese, "Conga: Distributed congestion-aware load balancing for datacenters," in *ACM SIGCOMM*, 2014.

[13] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," in *ACM SIGCOMM*, 2015.

[14] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *ACM SIGCOMM*, 2014.

[15] "Amazon EC2," https://aws.amazon.com/ec2/.

[16] "Microsoft Azure," https://azure.microsoft.com/en-us/.

[17] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, C. Kim, and A. Greenberg, "EyeQ: Practical network performance isolation at the edge," in *USENIX NSDI*, 2013.

[18] W. Wang and A.-L. Jin, "Friends or foes: Revisiting strategy-proofness in cloud network sharing," https://www.cse.ust.hk/~weiwa/papers/hug_plus.pdf, HKUST, Tech. Rep., 2016.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT Press and McGraw-Hill, 2009.

[20] "Amazon CloudWatch," https://aws.amazon.com/cloudwatch/.

[21] M. Chowdhury, "Coflow-Benchmark," https://goo.gl/szsBQE.

[22] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *ACM SIGCOMM*, 2011.

[23] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *ACM SIGCOMM*, 2015.

[24] M. Chowdhury, "Coflowsim," https://github.com/coflow/coflowsim.

[25] B. Briscoe, "Flow rate fairness: Dismantling a religion," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 63–74, 2007.

[26] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.

[27] S. Angel, H. Ballani, T. Karagiannis, G. O'Shea, and E. Thereska, "End-to-end performance isolation through virtual datacenters," in *USENIX OSDI*, 2014.

[28] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese, "Netshare and stochastic netshare: Predictable bandwidth allocation for data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 5–11, 2012.

[29] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multiresource allocation: Fairness–efficiency tradeoffs in a unifying framework," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1785–1798, 2013.

[30] A. Gutman and N. Nisan, "Fair allocation without trade," in *Proc. 11th Intl. Conf. Autonomous Agents and Multiagent Systems (AAMAS'12)*, 2012.

[31] J. Li and J. Xue, "Egalitarian division under leontief preferences," *Econ. Theory*, vol. 54, no. 3, pp. 597–622, 2013.

[32] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.