Towards flexible guarantees in Clouds: Adaptive bandwidth allocation and pricing

Dinil Mon Divakaran, Member, IEEE, and Mohan Gurusamy, Senior Member, IEEE

Abstract—This article focuses on the problem of bandwidth allocation to users of Cloud data centers. An interesting approach is to use advance bandwidth reservation. Such systems usually assume all requests demand either bandwidth-guarantee (BG) or time-guarantee (TG), but not both. Hence, the solutions are tailored for one type of requests. A BG request demands guarantee on bandwidth; whereas a TG request demands guarantee on time for transfer of data of specified volume. We define a new model that allows users to not only submit both kinds of requests, but also specify flexible demands. We tie up the problem of bandwidth allocation with differential pricing, that gives discounts to users based on the flexibility in their requests. We propose a two-phase, adaptive and flexible bandwidth allocator (A-FBA) that, in one phase admits and allocates minimal bandwidth to dynamically arriving user requests, and in another phase, allocates additional bandwidth for accepted requests maximizing revenue. The problem formulated in first phase is $\mathcal{NP}$-hard, while the second phase can be solved in polynomial time. We show that, in comparison to a traditional deterministic model, the A-FBA not only increases the number of accepted requests significantly, but also does so by generating higher revenues.

Index Terms—Cloud, bandwidth, QoS, reservation, pricing

1 INTRODUCTION

Bandwidth-sharing is a widely and deeply studied problem in networks. In the case of the Internet, the different design and deployment challenges are due to fairness in sharing, the heterogeneity of traffic classes and technologies, span of routes and paths across multiple autonomous networks having different administrative controls, etc. However, in the context of Clouds, the challenges form a smaller subset, thereby presenting an opportunity to explore known solution approaches, to share bandwidth between tenants. Importantly, a Cloud is usually under single administrative control (unlike the Internet), thereby making it easier to implement new policies. Besides, sharing policies have to take into account only a few number of fixed links that connect the Cloud data center to the Internet. The same argument is applicable to bandwidth-sharing within a data center, as the number of hops between servers is small and fixed.

A promising approach to guarantee performance for data communication is to use advance reservations. An admission-controller decides on the admission of every arriving request depending on the available bandwidth along the paths, and allocates the specified bandwidth for an accepted request respecting the constraints. However, such systems usually focus on one type of requests, either those demanding bandwidth-guarantees, or that requiring time-guarantees. The former, which we call bandwidth-guarantee (BG) requests, are those that are interested in specific values of bandwidth for given durations. The latter, referred here as time-guarantee (TG) requests, are those demanding transfer of data of specified volume within a given time-window. A Cloud that hosts multiple tenants and various applications can anytime get demands for both kinds of requests. For example, a VoD (video-on-demand) request for bandwidth-guarantee can exist along with a bulk data transfer request demanding guarantee only on time.

A major disadvantage in addressing just one type of requests is that the particular solution approaches (focusing on only one type) inherently lose the ability to exploit the other type of requests arriving in the system, thus affecting the revenue of providers. Such one-size-fits-all approaches also give few choices to users, thereby also affecting the customer base. For example, a system designed to allocate Cloud bandwidth to BG requests, forces users of TG requests to specify their request as bandwidth demands, while they are interested only in the transfer of data of specific volume before a deadline and not at the specific value(s) of bandwidth allocated.

Envisioning bandwidth-reservation capabilities in future data centers [7], [4], our work here takes a step towards user-centric approach for bandwidth allocation in Clouds. This work takes multiple deviations from the traditional assumptions on user requests:
1) We assume both kinds of requests—BG and TG—arrive at the (bandwidth-allocation) system.
2) We allow a user of BG request to specify a bandwidth-profile, instead of a single flat value. A bandwidth-profile gives bandwidth demands as a function of time.
3) The bandwidth-profile demanded by a user of a BG request need not be strict; rather it can be flexible within the bounds defined by the user. This also gives flexibility to a provider to allocate bandwidth within specified limits of the profile.
Observe how the third assumption introduces flexibilities for both users and providers. Past research works have often considered user demands to be strict, making it difficult for not only bandwidth allocations (as there is only one way of satisfying a demand), but also for user specification (as a flexible request can not be specified).

Our work makes the following contributions:

1) We develop a model that defines the parameters for describing BG and TG requests, as well as the conditions for accepting a request. The model allows users to specify flexible profiles for BG requests, at the same time giving flexibility to providers to allocate bandwidth.

2) As the cost incurred to users are dependent on the price of bandwidth, we tie up the model with pricing. We propose differential pricing as a way to induce incentives for users to specify flexible requests.

3) We develop a two-phase adaptive and flexible bandwidth allocator, called A-FBA, that processes dynamically arriving requests and decides on admitting them to the system. Each admitted request, be it a BG or TG request, is allocated bandwidth as per the conditions defined by the model.

The revenue generated by the A-FBA is dependent on the pricing model, which gives higher discounts to requests with higher flexibility. Our results show that the A-FBA is able to accept significantly higher number of requests than a traditional deterministic model that allocates flat rates to incoming requests. In addition, the A-FBA also generates higher revenue for providers, even when maximum discount is 20% of the price.

After discussing related works, Sec. 3 defines a model with BG and TG requests, defining the conditions for accepting a request. To solve the problem of bandwidth allocation, we first come up with the basic idea as a two-phase flexible bandwidth allocator (FBA) in Sec. 4, before developing A-FBA in Sec. 5. We introduce a new differential pricing scheme for the two kinds of requests in Sec. 6. Performance evaluation is carried out in Sec. 7.

2 RELATED WORKS

TG requests are typically the delay-tolerant bulk data transfer requests. Such huge amounts of data are generated not only by regular backup tasks, but also by Cloud service providers to improve end-to-end performance by replicating data across multiple data centers. The total volume of such transfers can range from terabytes to petabytes; see [10] and references therein. Large scientific experiments also generate huge amounts of data—CERN itself is known to generate petabytes of data annually that need to transferred to different data centers for scientific analyses [1]. In this context, Kosar et al. propose data transfer scheduling as a Cloud-hosted service [8].

Increasing number of works on bandwidth allocation assume bandwidth-reservation capabilities in future Cloud data centers [7], [4], [6], [9]; but they consider predetermined static rates for allocation. A predictive bandwidth auto-scaling system that reserves bandwidth from multiple data centers for VoD players was proposed recently [14]. The system aims to minimize reserved bandwidth. All these works consider only requests requiring bandwidth guarantees.

There are many applications that require time-varying bandwidth demands, in other words BG requests. VoD players might require different bandwidths at different times of the day. Similarly, users hosting their websites in a Cloud data center would have time-varying demands based on variations in, say, content, popularity, location of website visitors, etc. Within data centers, MapReduce jobs are known to have time-varying bandwidth demands; observing this, Xie et al. proposed an input abstraction consisting of a pre-defined set of profiles for requests, and a methodology for profiling traffic patterns to use the abstraction [18]. The work focussed purely on traffic demands (BG requests) within data centers. Whereas, our work considers both internal and external demands, BG and TG requests, as well as pricing as a mechanism for inducing flexibility in requests.

In our previous work [5], we proposed a model that allows a user to specify two bandwidth values—a minimum and a peak. Associated with the peak bandwidth are two probabilities that define a range for the fraction of the duration to allocate peak bandwidth. To an accepted request, the system allocates a bandwidth-profile that switches between the two bandwidth values, with peak bandwidth allocated for a time corresponding to the probability range. Hence, a bandwidth-profile was not part of the input; instead the system assigns it as output to an accepted request. Besides, the flexibility available to a provider was limited - at any time-slot the bandwidth in the profile would have to be either of the two values specified in input. Noting this disadvantage, our current work gives flexibility to the provider in choosing from infinite bandwidth values within the specified limits of the request. Besides, the work in [5] neither considered TG requests, nor used a pricing policy that offers incentives for flexibility in user requests.

There exists extensive literature on pricing in communication networks; for example [11, Ch. 22] discusses some important models. We refer the readers to a recent survey on broadband pricing models presented in [16], that classifies important policies into static and dynamic policies, as well as discusses pricing plans used in different countries. Moving away from the earlier convention of fixed flat-rate pricing of broadband data, large players like Amazon price Cloud resources following the usage-base (or the pay-as-you-go) model [2]. Aiming to decouple cost to tenants from the performance of the underlying network, Ballani et al. introduced a pricing mechanism in Clouds that is independent of the location of VMs of a tenant [3]. It ensures a minimum bandwidth to each tenant (based on its quota) and shares the spare bandwidth in proportion to tenants’ quotas.
The problem of pricing Cloud bandwidth for VoD providers was studied in [13]. The work argued on the benefits of multiplexing bandwidth reservations using a broker while providing performance guarantees. Observing the inability of tenants to accurately specify their requirements, Niu et al. studied the computationally challenging problem of determining the optimal policy for pricing Cloud bandwidth reservations under uncertainty [12]. They proposed a system which reserves a minimum bandwidth while allocating a fraction of the predicted demand dynamically with a certain risk. The focus of the work was to efficiently solve the optimization problems as well as to predict demand statistics. But we offload the prediction of traffic demands to users, and focus on the bandwidth allocation problem here.

We briefly discuss other related works in Sec. 1 of the online supplementary file.

3 Model description

We develop a bandwidth-reservation model here. A fraction of the link capacity is usually kept aside for multiplexing short-lived bursty flows, background traffic, etc., which may be priced using another model, say on a pay-as-you-go basis, as the bandwidth usage might be low. Hence the maximum bandwidth available for reservation on a link can possibly be less than the capacity. As is normal, reservation is useful for large, long-lived, bandwidth-consuming flows. Customers can also group a large number of flows originating from a (set of) VMs as a super-flow and demand bandwidth-reservation for the super-flow. For example, a web-hosting Cloud user requests for bandwidth to satisfy the aggregate traffic demands of his/her customers. In our model, we consider two types of requests: (i) bandwidth-guarantee (BG) requests, and (ii) time-guarantee (TG) requests.

BG requests are those that require certain amounts of bandwidth at different times. For example, a VoD provider requires specific amounts of bandwidth at different times. Besides, it is safe to assume that, with a prediction system, such users can specify the average approximate bandwidth values required at different times of the day, say normal and peak hours. A VoD player like Netflix can estimate the hourly bandwidth requirements from the Cloud for its users; it can then form a bandwidth-profile (defined in Sec. 3.1) to give as input to the reservation system, such that the flexibility permitted in the allocation of profile depends on the accuracy of the prediction system (the problem of using allocated bandwidth and other means, such as intelligent streaming applications, to provide guarantee to each customer of a VoD player, is out of scope of this work). For better utilization of allocated bandwidth, prediction has to be done either by the Cloud provider or by the user (say, Netflix); and in this work, we assume the users are better placed to estimate their demands.

A TG request is concerned about the transfer of a specific volume of data within a given time-window. The bandwidths assigned to a TG request at different times are not of interest to its user, but the aggregate bandwidth should be sufficient enough to transfer data within the time-window.

Remark: For simplicity, we abstract resource allocation to be over a single link. In Sec. 4 of the supplementary file, we discuss the case of multiple links and argue that the number of links that need to be considered for bandwidth allocation in a large-scale data center is generally limited to four.

3.1 Bandwidth-profile

Assume time to be fixed- and equal-length discrete slots. A bandwidth-profile gives values of bandwidth as a function of time. In Fig. 1, \( \lambda \) is an example of a bandwidth-profile requested by a user. A bandwidth-profile allocated (by the system) should be enforced at the end-hosts (or at entry points of the network) to guarantee performance. For practical purposes, we define bandwidth-profile of a request to be a step function over discrete time-slots. This is a common approach in the literature; e.g., [17] solves for step functions to perform deadline-constrained data transfers.

If the length of time-slots is too short, then TCP (due to slow convergence) will find it difficult to utilize the bandwidth allocated for the slot. But if time-slots are too long, the flexibility in bandwidth allocation reduces, as the bandwidth-profile might have only a few number of steps. Besides, having very long time-slots defeats the whole purpose of flexible allocation, as the problem then becomes closer to strict allocation of single rates to requests. We assume a time-slot to be in the range of tens of minutes to even an hour; but the right length will be decided by the provider depending on the applications used by the tenants. This would also mean, in the worst case, an accepted request may need to wait for one time-slot before starting to use its bandwidth-profile. Considering requests such as bandwidth for VoD players, transfer of bulk data, bandwidth for hosted websites, etc., the demands for which will be known well in advance, this is an acceptable delay.

For illustration purposes and for readability, we assume arriving requests are processed only once every time-slot. In reality, there can be an independent, different and shorter time-slot for processing of requests, as was done in our previous work [5]. That is, requests can be processed within much shorter durations.

3.2 Bandwidth-guarantee requests

We now define the parameters to describe a BG request \( r \).

- \( a_r \) is the arrival time of the request into the system,
- \( s_r \) denotes the start time of the request,
- \( e_r \) denotes the end time of the request,
- \( \lambda_r \) denotes the bandwidth-profile being demanded.

As \( \lambda_r \) is a step function over time-slots, \( \lambda_{r,\omega} \) denotes the bandwidth required by request \( r \) in time-slot \( \omega \).

- \( \sigma_r \) denotes the deviation factor that limits the permissible deviation of an allocated profile from the requested profile. \( 0 \leq \sigma_r \leq 1 \).
both providing time guarantee as defined above.

three time-slots, each of unit length. Fig. 2 illustrates two

We aim to provide two guarantees to an accepted BG request. If \( X_r \) is the bandwidth-profile allocated to an accepted BG request \( r \), then it should be such that,

\[
(1 - \sigma_r)X_{r,\omega} \leq X_{r,\omega}(1 + \sigma_r) \leq \lambda_{r,\omega}, \quad \forall \omega \in W_r, \tag{1}
\]

\( X_{r,\omega} \) being the bandwidth allocated for \( r \) in time-slot \( \omega \).

This condition mandates that the allocated bandwidth-profile should be between the limits specified by the input request. The flexibility in bandwidth-allocation available to a provider is dependent on the deviation factor. Higher the deviation factor, higher the flexibility.

An example of a requested bandwidth-profile and the corresponding allocated profile are given in Fig. 1. The requested profile alternates between two bandwidth values, \( b_1 \) and \( b_2 \). As illustrated, allocated bandwidth-profile \( X \) can take any values between the limits.

### 3.3 Time-guarantee requests

For a TG request \( r \), the input parameters include \( a_r, s_r, \) and \( e_r \), with the same meaning as for BG requests. In addition, \( v_r \) denotes the volume to be transferred. This means, for request \( r \in R^b \), the time-window during which the volume \( v_r \) should be transferred is \( W_r \), where \( W_r = \{s_r, s_r + 1, \ldots, e_r\} \). A user of an accepted TG request will also be allocated a bandwidth-profile, even though the user attaches no merit to the instantaneous values of bandwidth. If \( Y_r \) is the bandwidth-profile allocated to an accepted TG request \( r \), the guarantee that needs to be provided is,

\[
\sum_{\omega = s_r}^{e_r} Y_{r,\omega} = v_r \tag{2}
\]

For example, consider a TG request for transferring 2 * \( b_2 \) volumes of data between \( t_1 \) and \( t_4 \), i.e., within three time-slots, each of unit length. Fig. 2 illustrates two possible profiles that can be allocated for a TG request, both providing time guarantee as defined above.

### 4 Flexible Bandwidth Allocator (FBA)

We develop a two-phase flexible bandwidth allocator, or FBA in short. The FBA operates in two phases. Phase One decides on the admissibility of requests into the system, with an aim to maximize the revenue. This is achieved not by allocating as much bandwidth as possible to each accepted BG request, but on the contrary, by allocating the minimum bandwidth necessary to satisfy an accepted BG request as per the condition (1). Hence, Phase One accepts as many requests as possible. An accepted BG request obtains a (temporary) bandwidth-profile satisfying its demands in the request, at the same time giving more space for other requests. For accepting a TG request, the condition (2) has to be guaranteed.

Phase Two is executed after Phase One. Phase Two allocates bandwidth only for one slot, say \( \Omega \). It is run at a time as close to \( \Omega \) as possible so as to maximize the bandwidth allocated in that slot without affecting requests arriving in future. The allocator here picks up all accepted BG requests active in the time-slot \( \Omega \), and allocates as much additional bandwidth as possible to each request respecting condition (1), with an aim to maximize revenue due to allocation.

The phases will run towards the end of every time-slot processing requests for the following time-slot(s); and each taking a fraction of the slot-length for execution. In Sec. 2 of the online supplementary file, we motivate and illustrate the basic idea of FBA with an example.

#### 4.1 FBA: Formulation and algorithm

Let \( R^b \) and \( R^t \) denote the set of newly arrived BG and TG requests, respectively. Denote by \( p_{rb} \) and \( p_{rt} \), the price of (one unit of) bandwidth for a BG and a TG request, respectively. We describe the pricing scheme elaborately later, in Sec. 6. For now, it suffices to know that the pricing functions are linear. While pricing also allows us to have user budget as a constraint, we do not consider it in this work so as to focus on the problem of flexible bandwidth allocation. Besides, budget constraints can be roughly translated to bandwidth and flexibility requirements in a request.

The residual (available) bandwidth on the link is denoted by \( \rho_r \) with \( \rho_\omega \) being the residual bandwidth in time-slot \( \omega \). The duration \( d_r \) of a request \( r \) is obtained from the input: \( d_r = e_r - s_r + 1 \) (as \( s_r \) and \( e_r \) are assumed to be inclusive in the time-window of the request). Below we put down the formulations of the two phases. Sec. 4.1.3 gives the overall algorithm for the FBA, that binds both the phases. In Sec. 4.2, we discuss how this basic bandwidth allocation strategy can be improved.

#### 4.1.1 Phase One: Admission control

The revenue to optimize in Phase One is,

\[
\psi_1 = \sum_{r \in R^b} \sum_{\omega = s_r}^{e_r} p_{rb} X_{r,\omega} + \sum_{r \in R^t} \sum_{\omega = s_r}^{e_r} p_{rt} Y_{r,\omega} \tag{3}
\]
The variables $x_{r,s}$ and $y_{r,s}$ denote the bandwidth values allocated to a BG request and TG request, respectively, where the second subscript stands for time-slots of the requested window. We refer to $x_{r}$ and $y_{r}$ as the bandwidth-profiles allocated in Phase One. Define $R^{b}_{\omega}$ as the set of BG requests active in time-slot $\omega$:

$$R^{b}_{\omega} = \{ r | (r \in R^{b}) \land (\omega \in W_{r}) \}.$$ 

Similarly, $R^{t}_{\omega} = \{ r | (r \in R^{t}) \land (\omega \in W_{r}) \}$.

Denote by $T$ the union of the time-window sets ($W_{r}$’s) of all requests considered for scheduling during one execution of this phase. The mixed integer linear programming (MILP) formulation of Phase One optimization is:

$$\text{maximize } \psi_1$$

subject to:

$$x_{r,\omega} = \theta_{r}(1-\sigma_{r})\lambda_{r,\omega}, \quad \forall r \in R^{b}, \forall \omega \in W_{r} \quad (4a)$$

$$\sum_{\omega=s}^{t} y_{r,\omega} = \gamma_{r} v_{r}, \quad \forall r \in R^{t} \quad (4b)$$

$$0 \leq y_{r,\omega} \leq \gamma_{r} \rho_{\omega}, \quad \forall r \in R^{t}, \forall \omega \in W_{r} \quad (4c)$$

$$\sum_{r \in R^{b}_{\omega}} x_{r,\omega} + \sum_{r \in R^{t}_{\omega}} y_{r,\omega} \leq \rho_{\omega}, \quad \forall \omega \in T \quad (4d)$$

$$\theta_{r} \in \{0,1\}, \quad \forall r \in R^{b} \quad (4e)$$

$$\gamma_{r} \in \{0,1\}, \quad \forall r \in R^{t}. \quad (4f)$$

A rejected request $r$ will have its profile ($x_{r}$ for BG request, and $y_{r}$ for TG request) set to zero for the whole of its time-window. The first constraint is for BG requests. Recall that, for BG requests, the idea is to allocate only the minimum bandwidth required to meet the condition (1). The binary variable $\theta_{r}$, for a request $r$ is set to one if it is accepted, and zero otherwise (thereby ensuring zero bandwidth to rejected requests).

The constraints for allocating bandwidth to TG requests are straightforward: (i) constraint (4b) is nothing but the condition (2), i.e., sum of the bandwidth across the time-slots should be sufficient to transfer the requested volume, and (ii) the bandwidth can take any value limited by the residual link capacity (constraint (4c)). For a rejected TG request $r$, the binary variable $\gamma_{r}$ would be set to zero. Finally, constraint (4d) ensures that the link is not over-subscribed.

### 4.1.2 Phase Two: Dynamic Optimization

Phase Two allocates bandwidth in the following time-slot. For example, an execution in $\Omega$ is for allocating the residual bandwidth of slot $\Omega + 1$. As the accepted TG requests have been allocated the required bandwidth in Phase One, this phase considers only BG requests. To be precise, Phase Two processes only those set of accepted BG requests that are active in the time-slot $\Omega + 1$. If $A^{b}$ denotes the set of active (i.e., accepted and not yet terminated) BG requests, define the set $A^{b}_{\Omega+1}$ as:

$$A^{b}_{\Omega+1} = \{ r | (r \in A^{b}) \land (\Omega + 1 \in W_{r}) \}.$$ 

Let $\bar{x}_{r}$ denote the additional bandwidth allocated to request $r$ in this phase. The revenue to maximize is,

$$\psi_2 = \sum_{r \in A^{b}_{\Omega+1}} p_{r} \bar{x}_{r}. \quad (5)$$

Phase Two formulation can be written as,

$$\bar{x}_{r} + x_{r,\Omega+1} \leq (1+\sigma_{r})\lambda_{r,\Omega+1}, \quad \forall r \in A^{b}_{\Omega+1} \quad (6a)$$

$$\sum_{r \in A^{b}_{\Omega+1}} \bar{x}_{r} \leq \rho_{\Omega+1}. \quad (6b)$$

The first constraint ensures that the aggregate bandwidth assigned to a request in a slot is as per the request (condition (1)). The link capacity is respected with the second constraint.

#### 4.1.3 Algorithm for FBA

The online algorithm for allocating bandwidth-profiles to arriving requests is given in Algorithm 1. This algorithm is run towards the end of every time-slot, say $\Omega$, such that it completes in $\Omega$. $A^{b}$ maintains the set of BG requests active in the system, by regularly adding the accepted BG requests to $A^{b}$, as well as removing the terminating BG requests from $A^{b}$. The variable $Q^{b}_{\Omega}$ is used to denote the set of all accepted BG requests terminating in time-slot $\Omega$.

After executing each phase, the link bandwidth is updated based on the bandwidth allocated to BG and TG requests (line numbers 6 and 13).

#### Algorithm 1 FBA($R^{b}, R^{t}, A^{b}, \Omega$)

1: $R^{b}$: set of BG requests that arrived after previous scheduling instance
2: $R^{t}$: set of TG requests that arrived after previous scheduling instance
3: $A^{b}$: set of active BG requests (not terminated before time-slot $\Omega + 1$)
4: $\Omega$: time-slot (of execution)
5: Execute Phase One optimization on $R^{b}$ and $R^{t}$ (Sec. 4.1.1)
6: Update residual bandwidth $\rho$
7: $A^{b} \leftarrow A^{b} \setminus Q^{b}_{\Omega} \cup \{\text{newly accepted requests from } R^{b}\}$
8: $A^{b}_{\Omega+1} = \{ r | (r \in A^{b}) \land (\Omega + 1 \in W_{r}) \}$
9: Execute Phase Two optimization on $A^{b}_{\Omega+1}$ for the time-slot $\Omega + 1$ (Sec. 4.1.2)
10: for $r \in A^{b}_{\Omega+1}$ do
11: \hspace{1em} $X_{r,\Omega+1} \leftarrow x_{r,\Omega+1} + \bar{x}_{r}$ /* finalized bandwidth */
12: end for
13: Update residual bandwidth $\rho_{\Omega+1}$

### 4.2 Discussion

One disadvantage of the above algorithm is that it does not exploit the flexibility of requests requiring time-guarantees. A TG request is allocated a bandwidth-profile once it is accepted, which then becomes the final profile. This can potentially lead to rejection of other requests, which could have been incorporated if the overlapping TG requests were reallocated.

To illustrate, consider the example of two BG requests in Fig. 3: $r_{1}$ arrives at $t_{0}$, and $r_{2}$ at $t_{1}$. The bandwidth-profiles of both requests, $\lambda_{1}$ and $\lambda_{2}$, are given in Fig. 3,
As soon as the TG request is accepted, a transfer of some volume of data was already (possibly) done. Hence, to ensure that a previously accepted TG request does not cause rejection of a new request, we allow allocation; this indeed can lead to rejection of requests.

An adaptive allocator can accept all requests by allocating three units of bandwidth for each execution of the optimization phases, we denote the bandwidth-profile transferred for the TG request \( r \) (starting from \( \Omega + 1 \)). Clearly, \( \bar{W}_r \subset W_r \).

We are reallocating bandwidth-profiles of TG requests starting from \( \Omega + 1 \); therefore, bandwidths from time-slot \( \Omega + 1 \) that have been allocated previously, are added back to the link capacity before the execution of the phase.

Any reallocation of \( r \) (starting from \( \Omega + 1 \)) has to allocate bandwidth in the remaining time-slots (ending at \( \varepsilon^r \)) such that the aggregate is equal to \( \bar{\varepsilon}_r \). To achieve this, we add the following constraints to the list of constraints (4a)-(4f) given in Sec. 4.1.1:

\[
\sum_{\omega=\Omega+1} z_{r,\omega} = \bar{\varepsilon}_r, \quad \forall r \in A^f \tag{7a}
\]

\[
0 \leq z_{r,\omega} \leq \rho_{\omega}, \quad \forall r \in A^f, \forall \omega \in \bar{W}_r \tag{7b}
\]

\( z_{r,\omega} \) becomes the new bandwidth-profile for the TG request \( r \). The above constraints mandate allocation of sufficient bandwidth for data-transfer of an already accepted TG request. It can be observed that there is no need to change the maximization objective of the formulation. Next, replace the constraint (4d) in Phase One formulation of FBA on the link’s residual capacity, by,

\[
\sum_{r \in R^B_{\omega}} x_{r,\omega} + \sum_{r \in R^T_{\omega}} y_{r,\omega} + \sum_{r \in A^f_{\omega}} z_{r,\omega} \leq \rho_{\omega}, \quad \forall \omega \in T \tag{4d}
\]

where \( A^f_{\omega} = \{ r | (r \in A^f) \land (\omega \in W_r) \} \), is the set of accepted TG requests active in slot \( \omega \). Recall, \( R^B \) and \( R^T \) denote sets of newly arrived BG and TG requests; hence the intersection of any two of the three sets in the above equation \((x,y,z)\) is an empty set.

### 5.2 Phase Two

Consider the time-slot of interest when Phase Two is being executed (recall, this means the allocation is for \( \Omega + 1 \)). If there is unused residual capacity in time-slot \( \Omega + 1 \), then we can reallocate bandwidth for a TG request \( r \) that has been allocated bandwidth in a time-slot \( \omega > \Omega + 1 \); basically we advance the bandwidth allocated. This allows us to make space (bandwidth) for future requests. The remaining volume to be transferred, \( \bar{\varepsilon}_r \), is set to \( \bar{\varepsilon}_r = \sum_{\omega=\Omega+2} z_{r,\omega} \). The summation begins from \( \Omega + 2 \), as what we allocate in \( \Omega + 1 \) is in addition of the bandwidth already allocated for this slot in Phase One. If \( \bar{\varepsilon}_r \) is the bandwidth that will
be allocated in Phase Two for an active TG request \( r \) in the time-slot \( \Omega + 1 \). The revenue to optimize becomes,

\[
\psi_2 = \sum_{r \in A^b_{\Omega+1}} p^b_r \bar{x}_r + \sum_{r \in A^b_{\Omega+1}} p^r_r \bar{z}_r.
\]

For adapting the bandwidth allocated to active TG requests, we rewrite the (complete) optimization problem in Phase Two as:

\[
\text{maximize } \hat{\psi}_2
\]

\[
x_r - x_{r,\Omega+1} \leq (1 + \sigma_r) \lambda r, \Omega+1, \quad \forall r \in A^b_{\Omega+1} \\
0 \leq \bar{z}_r \leq \min((\hat{v}_r, \rho_{\Omega+1})), \quad \forall r \in A^b_{\Omega+1} \\
\sum_{r \in A^b_{\Omega+1}} \bar{x}_r + \sum_{r \in A^b_{\Omega+1}} \bar{z}_r \leq \rho_{\Omega+1}.
\]

After Phase Two execution, the variables \( \hat{v}_r \)'s are updated. In addition, for any non-zero bandwidth allocated to TG request \( r \) in \( \bar{z}_r \), we deduct the exact bandwidth from the farthest time-slot in \( \bar{z}_r \).

### 5.3 Algorithm for Adaptive-FBA

Algorithm 2 gives the steps for adaptive flexible bandwidth allocation, interweaving Phase One and Phase Two. \( Q^t_\Omega \) is the set of accepted TG requests terminating in slot \( \Omega \).

**Algorithm 2 A-FBA(\( R^p, R^t, A^b, A^t, \Omega \))**

1: \( R^p \): set of BG requests that arrived after previous scheduling instance
2: \( R^t \): set of TG requests that arrived after previous scheduling instance
3: \( A^b \): set of active BG requests
4: \( A^t \): set of active TG requests
5: \( \Omega \): time-slot index
6: \( \forall r \in R^t \). Update \( \hat{v}_r \)
7: Execute Phase One optimization on \( R^p, R^t \) and \( R^t_\Omega \) (Sec. 5.1)
8: Update residual bandwidth \( \rho \)
9: \( A^b \leftarrow A^b \cup \{ \text{newly accepted requests from } R^b \} \)
10: \( A^t \leftarrow A^t \cup \{ \text{newly accepted requests from } R^t \} \)
11: \( A^b_{\Omega+1} = \{ r | r \in A^b \text{ } & \text{ } (\Omega + 1 \in W_r) \} \)
12: \( A^t_{\Omega+1} = \{ r | r \in A^t \text{ } & \text{ } (\Omega + 1 \in W_r) \} \)
13: \( \forall r \in A^b_{\Omega+1}, \text{ Update } \hat{v}_r \)
14: Execute Phase Two optimization on \( A^b_{\Omega+1} \) and \( A^t_{\Omega+1} \) for the time-slot \( \Omega + 1 \) (Sec. 5.2)
15: for \( r \in A^b_{\Omega+1} \) do
16: \( x_r, \Omega+1 \leftarrow x_r, \Omega+1 + \bar{x}_r \text{ / finalized bandwidth } */
17: end for
18: for \( r \in A^t_{\Omega+1} \) do
19: \( Y_r, \Omega+1 \leftarrow Y_r, \Omega+1 + \bar{z}_r \text{ / finalized bandwidth } */
20: end for
21: Update residual bandwidth \( \rho_{\Omega+1} \)

### 5.4 Complexity analysis

Phase One of A-FBA is \( \mathcal{NP} \)-hard, while Phase Two can be solved in polynomial time. We show this in Sec. 3 of the online supplementary file. As described over there, Phase One can be reduced from the well-known multi-dimensional knapsack problem (MKP), an \( \mathcal{NP} \)-hard problem. There are interesting heuristics to efficiently solve the MKP. However, in this work we use an LP solver to solve both the optimization phases. In future, we plan to explore known heuristics for MKP and adapt one to solve the Phase One optimization problem.

Remark: The formulation and complexity analysis of A-FBA for the case of multiple links is given in Sec. 4.3 of the online supplementary file.

## 6 A DIFFERENTIAL PRICING OF BANDWIDTH

We describe how bandwidth can be differentially priced, so as to benefit both the provider and the users. From a provider’s point of view, pricing should help to increase the revenue as well as to induce incentives for flexible requests that will in turn help bandwidth allocation. For the users, pricing should give sufficient freedom to decide the best compromise between demand and cost. We devise a static pricing policy which (in essence) is similar to the Paris Metro Pricing (PMP) [15]. In PMP, the capacity is logically divided into separate bandwidth channels; these channels are priced differently, but are treated identically. The more expensive channels are less congested, thereby giving rise to differentiated performance to user traffic. In our case, channels correspond to flexibilities (for a given bandwidth).

For BG requests, the flexibility in allocating bandwidth depends on the deviation factor given as part of the request. If the deviation factor is zero, then the profile is strict in the sense that, the deviation of allocated bandwidth-profile from requested profile is strictly denied. So, if two requests demand the same bandwidth-profiles, but with different deviation factors, the user with a higher value of \( \sigma \) is giving higher flexibility to the provider than the user with a lower value. Hence, it would be appropriate to price bandwidth of a BG request based on, not only the absolute bandwidth demand, but also the deviation factor. As the deviation factor can take any value in the interval \([0, 1]\), we define the price of a unit of bandwidth for a BG request \( r \) as,

\[
p^b_r = k(1 - \delta \sigma_r),
\]

where \( k \) is a pre-determined maximum price, and \( \sigma_r \) is the deviation factor for request \( r \). This linear function permits a maximum discount of \( \delta k \), such that \( 0 \leq \delta \leq 1 \).

Similarly we argue that a data-transfer demand by a TG request becomes stricter as the average bandwidth required to transfer the data-volume gets close to a bandwidth-value (possibly the link capacity), say \( \bar{C} \). For this, we define a metric tightness, \( \tau_r \), for a request \( r \) as,

\[
\tau_r \equiv \frac{d_r}{\bar{C}},
\]

where \( d_r \) is the duration of (the time-window of) the request. Given the tightness, we define the price of a unit of bandwidth for a TG request as,

\[
p^t_r = k(1 - \delta(1 - \tau_r)),
\]

where \( k \) and \( \delta \) mean the same as before. Observe that the two price functions defined above for the two request
types are such that each allows the same maximum discount of δ fraction from the maximum price of k.
If the strictness of a BG request need not be equated to the same value of tightness of a TG request, the equations can be modified while keeping them still linear. Similarly, the maximum price of k can also be kept different for the two request types if needed.

7 Performance evaluation

We carry out performance evaluations in two stages. In the first stage (in Sec. 7.2), we compare three different bandwidth allocation strategies without pricing (i.e., \( p_r^b \) and \( p_r^l \) for all requests are set to one):

- Deterministic Bandwidth Allocator (DBA): This algorithm allocates a single flat bandwidth for an accepted request, and hence is a representative of the traditional bandwidth reservation systems.
- Flexible Bandwidth Allocator (FBA): The algorithm for FBA was described in Sec. 4.1.
- Adaptive Flexible Bandwidth Allocator (A-FBA), described in Sec. 5.

The second stage (Sec. 7.3) analyses the effect of differential pricing. Observing the advantage of A-FBA over FBA from the results of first stage, second stage compares DBA with A-FBA (and does not consider FBA).

7.1 Simulation setup

We describe the software and hardware systems used for simulation in Sec. 6.1 of the online supplementary file.

All values pertaining to input requests are generated randomly. Arrival times between requests, bandwidth values and durations of requests, all follow the Exponential distribution. The randomly generated bandwidth values (say, \( b_r \)’s) and duration are used to compute the volume to be transferred, in the case of a TG request, and the bandwidth-profile in the case of a BG request.

Each run generates 20,000 requests. For a single run, the mean duration of a request (be it TG or BG) is set as ten time-slots. The mean arrival rate of BG requests is set to 1.0. For TG requests, the mean rate is set relative to the mean arrival rate of BG requests, depending on the load. Given a specific value of input load for a single run, the mean bandwidth value is also estimated, and used to randomly generate bandwidth values (\( b_r \)’s) from the Exponential distribution.

For input to the DBA, both types of requests will have bandwidth demands (instead of bandwidth-profile and volume). That is, for each request \( r \) to FBA and A-FBA, there is a corresponding request \( r \) to DBA such that, \( a_r = a_r^F, s_r = s_r^F, c_r = c_r^F, \) and \( b_r^F \) is the bandwidth value. If \( r \) is a BG request, then \( b_r \) is set to the average of the bandwidth-profile, i.e., \( b_r = \frac{\sum_{r} \lambda_r \in \omega_r \cdot b_r^F}{\sum_{r} \lambda_r \in \omega_r} \). Instead if \( r \) is a TG request, \( b_r^F \) is \( \frac{v_r}{d_r} = b_r \). A bandwidth-profile, that forms part of a BG request, is generated using a Markov-Modulated Poisson Process (MMPP); we give the details in Sec. 5.2 on the online supplementary file.

Deviation factor \( \sigma_r \) of a request \( r \) takes a random value (uniformly) from the range \([0.1, 0.3]\). The link capacity

![Fig. 6. Comparing rejection percentage for varying loads](a) Scenario 1 (b) Scenario 2)

is 10 Gpbs. In each run, six settings differing in input loads (and not system loads) were considered—from 0.5 to 1.0—at steps of 0.1. For a given value of input load, the contributions due to BG requests and TG request is decided by a ratio that we define in the next section.

All figures here plot mean values from 20 runs (each with randomly generated 20,000 requests). The figures also mark 95% confidence intervals for the mean values.

7.2 Results - Stage 1: Comparison without pricing

To see how the loads due to the demands from the two requests types affect the results, we consider two scenarios differing in the ratio of loads due to BG and TG requests. The ratio is used to set, both, the number of BG and TG requests, as well as the bandwidth and volume demands due to BG and TG requests. Recall, the load is used to estimate the mean bandwidth of requests. In addition, the mean arrival rate of TG requests is dependent on load value. The two scenarios are:

**Scenario 1:** The ratio of loads due to BG and TG requests is 2:1. Hence, if the value of the input load is set to \( l \), then the contribution to the load due to bandwidth demands of BG requests will be \( \frac{2}{3} l \) and that due to volume demands of TG requests will be \( \frac{1}{3} l \). The mean arrival rate of TG requests is half that of BG requests. In addition, two-third of the 20,000 requests are BG requests, and the remaining one-third are TG requests.

**Scenario 2:** The ratio of loads due to BG and TG requests is 4:1. This also means, this scenario generates 16,000 BG requests and 4,000 TG requests. As in the previous scenario, the ratio of mean arrival rates too follows the load ratio.

Coming to the results, figures 6(a) and 6(b) give the percentage of requests rejected as a function of the input load, under the two scenarios. It can be seen that the FBA gives the worst performance, while A-FBA gives the minimum rejection ratio. This happens as the FBA does not exploit the flexibility of TG requests, and assigns a finalized bandwidth-profile to a TG request the very first time it is processed and accepted. Observe that the rejection percentage reduces for the FBA in Scenario 2 as the number of TG requests is relatively small.

The adaptive allocator A-FBA shows significant improvement over the other two strategies. The gain due to the adaptive policy is evident (in both the scenarios), with the A-FBA, in comparison with DBA, reducing the rejection percentage by more than 50% for all loads. The rejection percentage reduces by more than 80% for load values greater than or equal to 0.8, under both scenarios.
Fig. 7. Comparing rejection percentage of TG requests

Fig. 8. Comparing total bandwidth allocated

Fig. 9. Scenario where the ratio of BG to TG load is 1:4

Fig. 10. Maximum discount = 20%; ratio of loads (BG:TG) = 2:1.

We compare DBA and A-FBA, under this settings, in Fig. 9. Though A-FBA still reduces the rejection percentage considerably, the increase in bandwidth allocated is small. This setting is dominated by TG requests, and as the flexibility available to the A-FBA decreases with increasing tightness, expectedly, the total bandwidth allocated is only slightly higher than the DBA (≈ 4% on an average). The A-FBA still rejects much less number of requests than DBA, as A-FBA is split in phases such that the first phase allocates only the minimum required to accept a request, thereby leaving more bandwidth for future requests (to be accepted); whereas DBA allocates bandwidth in one step.

7.3 Results - Stage 2: Introducing differential pricing

Before discussing the results, we put down the values set for the different parameters governing the price. The price for a unit of bandwidth for both kinds of requests in the DBA was set to one unit. The value of k, that defines the maximum price for both kinds of requests under A-FBA (refer equations (9) and (11)), was similarly set to one. The maximum discount to any request is 20% of k, and hence δ was set to 0.2. We set the ratio of loads to 2:1, which means the demand due to BG requests is twice that of TG requests; besides, the ratio of number of BG requests was twice the number TG requests. The value of C was set to half the link capacity (see Eq. (10)). Total number of requests was 20,000 for each run. All other parameters influencing the generation of requests (following the input model) were set to the same values as in the previous section.

Fig. 10(a) plots the rejection ratio of requests as a function of the input load, under DBA and A-FBA. The significant reduction in rejection ratio brought about by the A-FBA is similar to that seen in the last section. This implies that pricing did not bring noticeable changes in the statistics of number of rejected requests.
Next we compare the revenue obtained from allocation of bandwidth in Fig. 10(b). With a maximum discount factor of 20%, the A-FBA was able to generate higher revenue from accepted requests than the DBA. The percentage increases in revenue for loads of 0.5 to 1.0 were approximately 9 to 17, with an average increase of ≈12%. Even for low to moderate loads, approximately 8-10% higher revenue was generated with the A-FBA.

Fig. 11 plots the bandwidth allocated by the two allocation strategies. The plot for total bandwidth allocated and revenue generated is same for the DBA, as the price was set to one unit. We see that the A-FBA allocated ≈18% to 26% (for loads from 0.5 to 1.0) more bandwidth than DBA. While the average increase in revenue was ≈12%, the average increase in bandwidth allocated was ≈21%. Recall, in this set of experiments the maximum discount was 20%. A provider can decrease the maximum discount to bring in higher revenues. For example, with 15% maximum discount, A-FBA achieved an average increase of ≈14% in revenue, for an average increase of ≈21% in bandwidth allocated (over DBA).

Fig. 12 plots the total revenue generated for different values of maximum discount (from 5% to 40%). We chose a medium load of 0.7 for this experiment; and the load due to BG requests was twice that of TG requests. The increase in revenue due to A-FBA was maximum at ≈18% for discount not more than 5%, and this decreases with increasing discounts. In another set of simulations, where the load due to TG requests is four times that of BG requests, we observed that the maximum discount has to be reduced to approximately 8% (of the price), for A-FBA to have a revenue not less than that of DBA.

### 7.4 Discussion

The results demonstrate that A-FBA decreases rejection ratio while generating higher revenue, in comparison to the traditional deterministic bandwidth allocator (DBA), even when the deviation factor is low. Similarly, with a discount of up to 20% of the price, A-FBA generates much higher revenue than DBA. We also conducted experiments to understand the effect of increasing flexibility on rejection ratio and revenue. The results are presented in Sec. 5.3 of the online supplementary file. We observed that with A-FBA, higher flexibility in user requests reduces rejection ratio even further. A-FBA also generates higher revenue than DBA, though our pricing policy offers lower price for higher flexibility for the same bandwidth demand.

### 8 Conclusions

This work defined a new model for describing two important types of requests—bandwidth-guaranteeing and time-guaranteeing—in the context of Cloud data centers. The model allows users to specify flexible requests, while allowing providers to exploit the mix of two kinds of requests with flexible demands. We introduced a differential pricing mechanism for giving discounts to users based on the flexibility in the requests. We developed a two-phase adaptive flexible bandwidth allocator that decides on the admissibility of dynamically arriving requests and allocates bandwidth with the objective of maximizing revenue for providers. Our study revealed that, in comparison to the deterministic bandwidth allocator, A-FBA not only significantly increases the number of accepted requests, but also generates higher revenues.

We discuss some interesting directions for future works in Sec. 6 of the online supplementary file.

### References

**Dinil Mon Divakaran** is a Scientist at the Cybercrime & Security Intelligence department in the Institute for Infocomm Research (I2R), Singapore. Prior to this, for two years he was a Research Fellow at the Department of Electrical and Computer Engineering in the National University of Singapore (NUS). He has also worked as an assistant professor in the School of Computing and Electrical Engineering at the Indian Institute of Technology (IIT) Mandi. He carried out his PhD at the INRIA team in ENS Lyon, France. He holds a Master degree from IIT Madras, India. His research works revolve around applications of game theory, queueing theory and probabilistic models, as well as the study of optimization problems and design of heuristics, all in the broad area of computer networks. He is keenly interested in the study of architectures and protocols in the context of network security and QoS delivery.

**Mohan Gurusamy (M'00-SM'07)** received the Ph.D. degree in Computer Science and Engineering from the Indian Institute of Technology, Madras in 2000. He joined the National University of Singapore in June 2000, where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. He is currently serving as an editor for Elsevier Computer Networks journal and Springer Photonic Network Communications journal. He has served as the lead guest editor for two special issues of the IEEE Communications Magazine (OCS), August 2005 and November 2005, and as a co-guest editor for a special issue of the Elsevier Optical Switching and Networking journal, November 2008. He was the organizer and lead chair for CreateNet GOSP workshops co-located with Broadnets conference, October 2005 and October 2006, and September 2008. His research interests are in the areas of Data Center networks, optical networks, Metro/Carrier Ethernet networks and wireless sensor networks. He has over 135 publications to his credit including two books and three book chapters in the area of optical networks. Dr. Gurusamy has been a member of the IEEE since 2000 and senior member since 2007.