IoNCloud: exploring application affinity to improve utilization and predictability in datacenters

Daniel S. Marcon*, Miguel C. Neves*, Rodrigo R. Oliveira*, Leonardo R. Bays*,
Raouf Boutaba†, Luciano P. Gaspary*, Marinho P. Barcellos*
* Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
† David R. Cheriton School of Computer Science, University of Waterloo, Canada
{daniel.stefani, mcneves, ruas.oliveira, lrbays, paschoal, marinho}@inf.ufrgs.br, rboutaba@uwaterloo.ca

***Abstract*—The intra-cloud network is typically shared in a best-effort manner, which causes tenant applications to have no actual bandwidth guarantees. Recent proposals address this issue either by statically reserving a slice of the physical infrastructure for each application or by providing proportional sharing among flows. The former approach results in overprovisioned network resources, while the latter requires substantial management overhead. In this paper, we introduce a resource allocation strategy that aims at providing an efficient way to predictably share bandwidth among applications and at minimizing resource underutilization while maintaining low management overhead. To demonstrate the benefits of the strategy, we develop IoNCloud, a system that implements the proposed allocation scheme. IoNCloud employs the abstraction of attraction/repulsion among applications according to their temporal bandwidth demands in order to group them in virtual networks. In doing so, we explore the trade-off between high resource utilization (which is desired by providers to achieve economies of scale) and strict network guarantees (necessary for tenants to run jobs predictably). Evaluation results show that IoNCloud can (a) provide predictable network sharing; and (b) reduce allocated bandwidth, resource underutilization and management overhead when compared against state-of-the-art proposals.**

I. Introduction

Cloud providers lack practical, efficient and reliable mechanisms to offer bandwidth guarantees for applications [1], [2]. The intra-cloud network is typically oversubscribed and shared in a best-effort manner, relying on TCP to achieve high network utilization and scalability. TCP, nonetheless, does not provide robust isolation among flows in the network [3]; in fact, long-lived flows with a large number of packets are privileged over small ones (which is typically called *performance interference* [4]) [5]. Moreover, recent studies [6], [7] show that bandwidth available for virtual machines (VMs) in the intra-cloud network can vary by a factor of five or more, resulting in poor and unpredictable overall application performance.

The lack of network guarantees directly impacts both tenants and providers. Tenants are unable to enforce the allocation of network resources for their requests (which particularly hinders applications with strict bandwidth requirements) and can only deploy some specific enterprise applications in the cloud [8]. Moreover, costs are unpredictable due to high network variability (in many services, the subsequent computation depends on the data received from the network [9], [10]). Providers, in turn, may lose revenue, because performance interference ends up reducing datacenter throughput [1], [6].

Recent proposals [3], [6], [8], [11], [12] address this issue either by offering minimum guarantees or by providing proportional sharing. The former explicitly reserves a slice of the physical infrastructure for each application, which results in overprovisioned resources for tenants (since the temporal network usage of applications is not constant). The latter, in turn, assigns administrator-specific weights for entities (such as VMs and processes) in order to provide proportional sharing at flow-level in the network. However, it requires substantial management overhead, since bandwidth consumed by each flow is determined according to its weight for each link in the path, and large-scale datacenter networks can have over 10 million flows per second [13].

In this paper, we leverage the key observation that temporal bandwidth demands of cloud applications do not peak at exactly the same time [14], [15] and propose a resource allocation strategy for reserving and isolating network resources in cloud datacenters. It aims at minimizing resource underutilization while providing an efficient way to predictably share bandwidth among applications, with low management overhead. To show the benefits of the strategy, we develop IoNCloud (Isolation of Networks in the Cloud), a system that implements the proposed allocation scheme. IoNCloud employs the abstraction of attraction/repulsion among tenant applications according to their temporal network usage and need of isolation, and groups them into virtual networks (VNs) with bandwidth guarantees. In doing so, we seek to explore the trade-off between high resource utilization (a common goal for providers to reduce operational costs) and strict network guarantees (desired by tenants).

Overall, the major contributions of this paper are threefold. First, we propose a topology-agnostic network-performance-aware resource allocation strategy for cloud datacenters. It improves network predictability by grouping tenant applications into virtual networks according to their temporal bandwidth demands. Second, we develop IoNCloud, a system that implements the proposed strategy for large-scale datacenters. IoNCloud (i) groups applications in VNs; (ii) maps them on the physical substrate; and (iii) provisions network resources at each link the VN was allocated on according to peak aggregate demands of applications in the same group that utilize the link (i.e., the bandwidth required at the period of time when the sum of network demands of applications belonging to the same group is the highest). Third, we evaluate and show that, in comparison with the state-of-the-art [6], IoNCloud provides the same level of network predictability with less bandwidth reserved for applications, reduced resource underutilization and lower management overhead.

The remainder of this paper is organized as follows.

Section II examines related work. In Section III, we introduce our resource allocation strategy (and its implementation, IoNCloud), and Section IV presents the evaluation of the proposed strategy. Finally, Section V discusses the generality and limitations of IoNCloud, and Section VI closes the paper.

II. Related Work

Most related approaches attempt to offer bandwidth guarantees by taking advantage of rate-limiting at hypervisors, VM placement and VN embedding in order to increase their robustness. They can be separated in three classes, as discussed below.

Spatial-temporal awareness. Proteus [9], Choreo [16] and the approach developed by Chen and Shen [15] use spatial and temporal demands of applications to map them in the cloud. However, they present some drawbacks. Proteus requires a complex allocation scheme and provides only a rigid network model for each application, defined at its allocation time. Choreo requires its placement algorithm to have detailed knowledge about current network state; such information may not be easily obtained in large-scale datacenter networks. In particular, Proteus and Choreo may add substantial management overhead to achieve their goals. Finally, Chen and Shen only focus on temporal demands of computing resources.

Network guarantees. Oktopus [6] and SecondNet [17] provide strict bandwidth guarantees by isolating each application in a distinct VN. Despite their benefits, these approaches result in underutilization of resources and internal fragmentation of both computing and network resources upon high rate of tenant arrival and departure. EyeQ [12] and Gatekeeper [18], in turn, can offer bandwidth guarantees only when the core of the network is congestion-free. Baraat [1] and Varys [10] achieve high network utilization, but cannot provide strict bandwidth guarantees for tenants. Finally, ElasticSwitch [8] and the Logistic Model [3] are orthogonal to our approach, as they assume there exists an allocation method in the cloud platform (i.e., applications are already allocated).

Proportional sharing. Seawall [4] and Hadrian [11] allow bandwidth sharing at link-level according to weights assigned to VMs. FairCloud [19], in turn, proposes mechanisms that explore the trade-off among network proportionality, minimum guarantees and high utilization. These methods, however, result in substantial management overhead, because bandwidth consumed by each flow at each link is determined according to its weight in comparison to other flows sharing the same link (and the intra-cloud network can have over 10 million flows per second [13]).

In summary, these approaches either result in overprovisioned network resources or require substantial management overhead. Therefore, we introduce a resource allocation strategy that aims at providing network predictability with reduced bandwidth underutilization and low management overhead, and materialize it by developing a system called IoNCloud.

III. IoNCloud

The IoNCloud system implements our novel approach to allocate tenant applications in large-scale cloud platforms. The proposed strategy aims at providing performance predictability in the intra-cloud network while minimizing resource underutilization. To achieve this, unlike previous work [6], [9], [17], [18], IoNCloud groups applications in virtual networks according to their temporal bandwidth demands.

In this strategy, all applications that belong to the same group share the same set of (virtual) network resources (i.e., they have shared bandwidth guarantees). Virtual networks, in turn, are completely isolated from one another, which means that each group has a guaranteed amount of network resources. An abstract view of IoNCloud is shown in Figure 1, which depicts application requests being received and allocated in two steps. The first step is responsible for application demand analysis and grouping, while the second maps VNs (groups composed of sets of applications) onto the physical substrate.

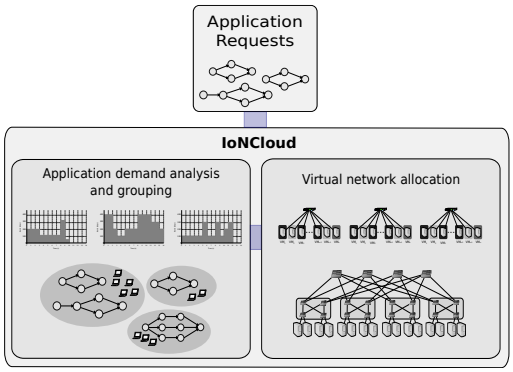


Fig. 1: IoNCloud model overview.

We first discuss how to obtain network profiles of applications (Section III-A) and describe application requests (Section III-B). Then, we present our novel strategy to group complementary applications in VNs (Section III-C) and to embed VNs on the cloud substrate (Section III-D).

A. Network Profile of Applications

IoNCloud assumes that network profiles were previously generated (using techniques proposed in the literature, such as the ones described by Xie et al. [9], LaCurts et al. [16] and Lee et al. [2]) and, thus, uses such information as input for incoming application requests. Like Choreo [16], IoNCloud considers, in each profile, the number of bytes sent by the application rather than the observed rate, as the former is independent of network congestion.

In particular, we highlight the feasibility of obtaining these profiles. Several studies [6], [9], [13], [15], [16] have conducted experiments on VM resource utilizations during both short- and long-term periods of time. Their main findings are summarized as follows: i) application traffic patterns are predictable; ii) VMs of the same application (such as MapReduce) tend to have similar resource consumption; iii) the same application running different datasets has similar patterns of resource usage; and iv) periodical (e.g., daily) patterns of resource utilization can be observed for long-term applications.

These results enabled the proposal of some techniques to profile cloud applications. For instance, Xie et al. [9] and LaCurts et al. [16] use network monitoring tools (sFlow and tcpdump) to collect traffic traces (gathering application communication patterns), while Lee et al. [2] discuss the utilization of application templates (provided as a library for users). Xie et

al. also convert the output of these measurements into coarse-grained pulse functions. Both studies perform these profiling runs during a testing phase or in production environments, which allows them to collect sufficient information to create network profiles before running applications in the cloud. Therefore, such techniques can be used for IoNCloud, so that application profiles are known before allocation.

IoNCloud also considers applications that cannot have their network profiles generated in advance (for instance, because the application requires an elevated amount of resources to run a profiling test or concludes very quickly). In such situations, the time-varying function $B(t)$ (detailed in Section III-B), which indicates the temporal network demands of applications, is represented by a constant function (i.e., the same bandwidth requirement during the entire application lifetime). This constant value is specified by the tenant when submitting the request to the cloud.

B. Application Requests

Tenants request applications using the hose-model (similarly to prior work [8], [9], [19], [20]). In this abstraction, all VMs of an application are connected to a virtual switch through dedicated bidirectional links. Each application request is represented by its resource demand and formally defined by <N, $B(t)$>, with the terms specifying the number of VMs and the temporal bandwidth required by the application. The bandwidth demand is a time-varying function $B(t)$, similarly to Proteus [9]. It represents the bandwidth required by the application at time "t". The amount of bandwidth of each link connecting a VM to the application virtual switch is represented by $\max(B(t))$, which denotes the peak temporal demand of the application's VM. This fine-grained specification allows IoNCloud to capture network requirements of applications in a precise manner.

Without loss of generality, we follow previous work [6], [9] and make two assumptions. First, we abstract away computing resources and assume a homogeneous set of VMs (i.e., equal in terms of CPU, memory and storage consumption). Second, we consider that all VMs of a given application follow the same network model[1].

C. Application Demand Analysis and Grouping

This first step is responsible for analyzing network demands of applications and grouping them in VNs. This way, high resource utilization can be achieved without hurting predictability.

Figure 2 shows an example of how IoNCloud performs this process. In Figure 2(a), bandwidth requirements (dashed lines) of two applications (A and B) are fully guaranteed through a simple static reservation model (where the peak bandwidth is reserved, represented by dotted lines). However, this basic model causes underutilization of resources (shown by arrows in the figure), as unused bandwidth of one application (virtual network) cannot be used by any other application [8], [9]. IoNCloud, in contrast, enables applications with complementary bandwidth requirements to share network resources. This is done by grouping them in the same VN and reserving the peak aggregate demand, represented by the dotted line in Figure 2(b). Therefore, IoNCloud achieves better resource utilization, since periods of low demand from one application can be compensated by periods of high demand from other ones. Note that IoNCloud removes performance interference in the network by reserving the peak aggregate bandwidth of the applications (that belong to the same group) sharing a given link. Furthermore, it significantly reduces management overhead when compared to Proteus [9], since the latter must modify reservations as time passes by.

Algorithm. The key idea is based on minimizing the amount of unused bandwidth for each group created (i.e., reducing wasted bandwidth). Algorithm 1 retrieves one application (app) at a time from the set of applications $Applications$ and verifies three possibilities of grouping: i) creating a new group composed of app and another application from the set $Applications$ (i.e., trying all pairs of possible groupings of app with other incoming applications and selecting the one with least underutilization); ii) inserting app into one of the existing groups; and iii) creating a new group with app only. After verifying these possibilities, the best option is selected. Finally, once the entire bundle of incoming applications has been analyzed and included into groups, the algorithm concludes, returns the set of groups and the process of allocating each group (represented by a VN) on the cloud is started.

Algorithm 1: Network-aware group creation.

Input : Bundle of applications to be allocated in the cloud
Output: List of application groups $GroupList$

```
Create a set Applications with all incoming applications;
Create an empty set GroupList;
foreach app ∈ Applications do
    Evaluate three possibilities of grouping:
        Creating new group containing app and a chosen application from Applications;
        Including app in existing group of the set GroupList;
        Creating new group with single application app;
    Among the three above, select the option with least underutilization;
    Remove grouped applications from Applications;
    if new group was created then
        Include new group in the set GroupList;
return the set GroupList;
```

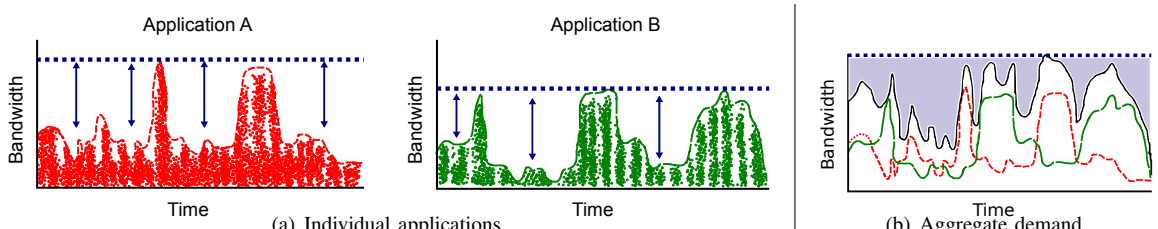
D. Virtual Network Allocation

This step is responsible for allocating each VN (group composed of applications that present complementary temporal bandwidth demands) on the physical infrastructure.

A simplified example is shown in Figure 3, where there are only two groups to be allocated, each one with two applications. For each group, IoNCloud prioritizes clustering VMs of the same application in the same physical server, since good locality reduces the amount of network resources used for intra-application communication[2]. For a single application, VMs located in the same server do not consume network resources when they communicate with each other. VMs allocated in different servers, in turn, need a certain amount of bandwidth to exchange data, which is given by the server with the lowest peak aggregate rate for an application. Consider "Server 1" (S_1) and "Server 3" (S_3) in the figure, which host application 1 (app_1). The bandwidth needed by VMs of this

[1]Many applications, such as MapReduce (which represents an important class of applications running in datacenters), have similar bandwidth demands among their VMs [9].

[2]We follow related work [6], [9], [18] and consider only intra-application communication when allocating applications in the cloud platform, as this type of communication represents most of the traffic in the cloud [11].



(a) Individual applications. (b) Aggregate demand.

Fig. 2: Temporal network usage.

application for communication among themselves is given by $\min(|S_{1,app_1}|, |S_{3,app_1}|) * \max(B(t))$, where $|S_{x,app_1}|$ represents the number of VMs of app_1 placed at the x^{th} server and $\max(B(t))$ denotes the peak bandwidth used by a single VM during its lifetime. In this example, $\min(6, 2) * 15 = 30$ Mbps.

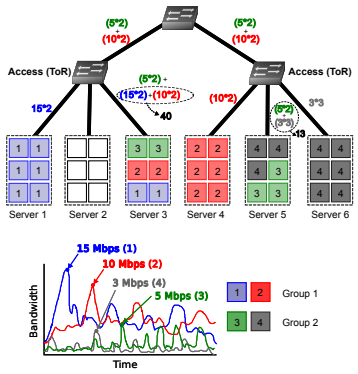


Fig. 3: Shared bandwidth guarantees for applications.

A group of applications, in contrast, requires the peak aggregate bandwidth demand of the group. Therefore, the allocated bandwidth on each physical link l of a VN corresponds to the peak aggregate demand of VMs in the group that use l. This allocation is illustrated in Figure 3 in two situations: i) when more than one application of the same group share a link, the aggregate peak bandwidth of the VMs of these applications is reserved: for instance, in Server 3, VMs of applications 1 and 2 from group 1 share the access link and, if they were isolated, they would require 50 Mbps ($15 \times 2 + 10 \times 2$), but only 40 Mbps is reserved because this is the peak temporal demand of the group for this link (according to the temporal demands of applications shown below the servers in the figure); and ii) when a single application of a group uses a link, the bandwidth reserved corresponds to the amount needed by its VMs alone (other applications of the group do not need to use that link, as we can see in core links). Last but not least, note that VNs do not share bandwidth with one another (i.e., groups are completely isolated).

Algorithm. Algorithm 2 takes advantage of application affinity to allocate VNs on the substrate[3]. Since locality is key to make efficient use of resources, we address it with two granularities: i) VMs of the same application are mapped on the infrastructure according to a VM placement objective (since IoNCloud is agnostic about VM placement, these objectives will be detailed after the overview of the allocation algorithm); and ii) VMs belonging to applications of the same group are allocated close to each other, because their bandwidth demands are complementary and, thus, network underutilization can be reduced (as determined by the grouping algorithm in the previous step). The algorithm allocates one VN at a time, with a coordinated node and link mapping, following insights provided by Chowdhury et al. [21]. The first step is the allocation of nodes (VMs) for each application in the group, according to the VM placement policy defined. After all VMs of a VN are allocated, the algorithm starts the second step of the mapping, which is the allocation of bandwidth for the group according to the example shown in Figure 3. The algorithm returns a success code for each VN that was embedded on the substrate and a failure code otherwise.

[3]Like related work [6], [9], [18], [20], we assume the physical infrastructure topology in cloud datacenters is defined as a multi-rooted tree [5].

Algorithm 2: Virtual network embedding.

Input : Physical infrastructure P, Set of groups $GroupList$
Output: Success/Failure array $allocated$

```
foreach Group g ∈ GroupList do
    // VM allocation
    foreach Application app ∈ g do
        Allocate VMs of app in the cloud infrastructure according to a
        predefined objective (e.g., minimum bandwidth, energy consumption,
        or fault tolerance);
    // Bandwidth allocation
    foreach Level lv from 0 to Height(P) do
        Allocate bandwidth at lv according to demands of VMs at lower
        levels (similarly to Figure 3);
    allocated[g] ← success/failure code for the allocation of group g;
return allocated;
```

VM placement objectives. VM placement is often implemented as a multi-dimensional packing with constraints being defined according to a placement goal. IoNCloud currently supports three different goals, as follows. First, *MinBand* minimizes bandwidth consumption by clustering VMs of the same application and of the same group on the smallest subtree in the physical infrastructure (similarly to Ballani et al. [6]). Second, *MinEnergy* follows insights from Mann et al. [22] and uses a first-fit algorithm to reduce the number of servers turned on, thus minimizing the total amount of power consumed by these servers. Third, *MaxFT* considers fault tolerance by spreading VMs on the cloud platform, so that applications can survive upon link, switch and/or rack failures (similarly to Bodík et al. [23]). The key idea is to increase the number of servers used to allocate VMs in accordance to a given spreading factor (*sf*). In particular, the minimum number of servers is determined considering the servers with available resources, and the number of VMs from the application each one of them can host. The new number of servers that will host these VMs is determined by choosing the minimum value

between (i) the multiplication of the minimum number of servers required to allocate such VMs and sf and (ii) the number of VMs of the application: $\texttt{ExpectedNumSrvs} = \min(\texttt{MinNumSrvs}(\texttt{App}) * sf, \texttt{NumVMs}(\texttt{App}))$.

Allocation quality. Algorithm 2 was designed as a constructive heuristic with the focus of providing efficient allocation of resources. It does not consider optimality, because it is computationally expensive to employ optimization strategies for large-scale cloud platforms [6], [20] and the allocation must be performed as quickly as possible (since there are high rates of tenant arrival and departure [4], known as churn). We defer a detailed study of the advantages and drawbacks of employing optimization models for IoNCloud to future work.

IV. Evaluation

In this section, we demonstrate the benefits of IoNCloud for both providers and tenants. Our evaluation focuses primarily on quantifying the advantage of grouping applications in virtual networks in terms of network predictability and resource utilization. Toward this end, we first describe the environment and, then, present the main results.

A. Environment

Datacenter topology. We follow previous work [6], [9], [20] and implement a discrete-event simulator that models a multi-tenant datacenter. We focus on tree-like topologies similar to multi-rooted trees used in current cloud platforms [11]. The physical substrate is defined as a three-level tree topology with 8,000 servers at level 0, each with 4 VM slots (i.e., with a total amount of 32,000 available VMs in the cloud). Every machine is linked to a ToR switch (40 machines form a rack), and every 20 ToRs are connected to an aggregation switch. Finally, all aggregation switches are connected to a core switch. Link capacities are defined as follows: machines are connected to ToR switches with access links of 1 Gbps; links from racks up to aggregation switches are 10 Gbps; and aggregation switches are attached to a core switch with links of 50 Gbps. Thus, the default oversubscription of the network is 4.

Workload. In line with related work [6], [9], [11], we generated the workload according to results obtained by measurement studies [4], [13], [24]. More specifically, the workload is composed of requests of applications to be allocated in the cloud platform. Requests are formed by a heterogeneous set of applications (including MapReduce and Web Services), which is representative of applications running on public cloud platforms [5]. Each application is represented as a tuple <N, $B(t)$>, with N being the number of VMs and $B(t)$ a time-varying function to specify the temporal network demand. The former is exponentially distributed around a mean of 49 VMs (representative of current clouds [4]). The latter was generated following results obtained by Benson et al. [13] and Kandula et al. [24] (we used measurements related to inter-arrival flow time and size at servers to simulate application traffic).

B. Results

We compare IoNCloud, which employs shared bandwidth guarantees, with the approach adopted by most related work [6], [17], [18], which creates one virtual network per application. Ideally, we would have compared IoNCloud with Proteus [9]. Proteus uses as input pulse functions obtained from the temporal network demands of applications. However, the generation of such pulse functions is addressed as a black-box in the paper and, thus, we cannot precisely develop a generator that mimics its behavior.

As previously mentioned (Section III-D), the algorithm used for virtual network allocation is agnostic in terms of VM placement. Hence, three VM placement algorithms are used in experiments: (i) *MinBand*, which minimizes the amount of bandwidth reserved for communication between VMs; (ii) *MinEnergy*, which minimizes energy consumption by reducing the number of used servers; and (iii) *MaxFT*, which maximizes fault-tolerance based on a given parameter (the desired ratio of extra servers used for spreading VMs).

For all experiments, we plot the percentile difference between both approaches given by the following equation: $(\frac{\texttt{IoNCloud}}{\texttt{One VN per App}} - 1) * 100\%$. Hence, negative percentiles mean IoNCloud has achieved a lower value than traditional approaches, while positive percentiles mean IoNCloud has achieved a higher value than traditional approaches. In general lower values are better, with the sole exception being Figure 7.

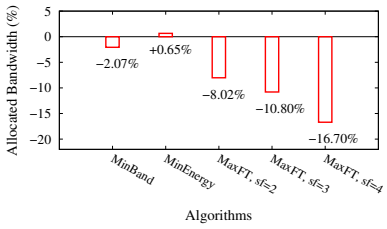


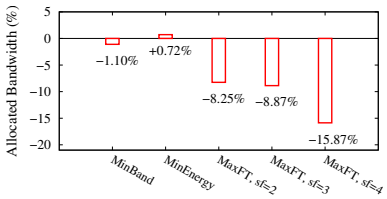
Fig. 4: Amount of reserved network resources.

Amount of reserved network resources. Figure 4 shows the total amount of reserved network resources according to the different placement algorithms. The y-axis represents the percentile difference between both approaches regarding the amount of bandwidth allocated, hence *the lower the value, the better*. We see that for any given approach, the amount of reserved resources increases in accordance with VM spreading. As expected, the shared bandwidth mechanism employed by IoNCloud outperforms the traditional methods when VMs are spread around the network, as it reduces the amount of reserved resources (up to 16.70%). This means that the provider can accept more applications in the cloud, improve resource utilization and, ultimately, increase datacenter throughput.

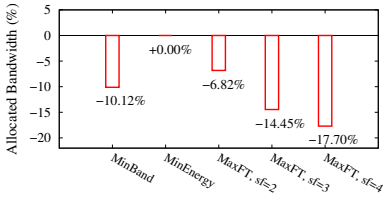
In contrast, IoNCloud is unable to achieve gains (in fact, with 0.65% of overhead in the worst-case) when there is no spreading, that is, when VMs are as packed as possible. This happens because the resource reservation employed by IoNCloud is performed per group, instead of per application (as traditional approaches). Therefore, the bandwidth allocated to each virtual link is only released after all applications in the respective group have finished. This design choice was deliberately chosen; such model can reduce the overhead of calculating the amount of bandwidth to be deallocated for each application that finishes its execution at each virtual link of the group. Moreover, we expect VM spreading to be norm in real cloud networks due to the high churn [4] of applications in these environments.

We further analyze bandwidth allocation by measuring the

amount of reserved resources in access and aggregation levels[4] of the topology for all VM placement algorithms. We see in Figure 5 that IoNCloud allocates less resources at both levels. In particular, note that IoNCloud has better results in the aggregation. This effect also increases the chance of allocating virtual links, since network oversubscription at this level is higher than at the edge, and decreases the probability of packet discards in the network (which usually happens at this level [13]).



(a) At the access.



(b) At the aggregation.

Fig. 5: Per-level analysis of reserved network resources.

Underutilization in the network. Figure 6 depicts the percentile difference of unused bandwidth for different placement algorithms. Underutilization is quantified by measuring the unused bandwidth on each virtual link. *Lower values are better*, since they mean that the cloud infrastructure is making better use of its reserved resources. As expected, IoNCloud achieves lower underutilization than current approaches. In fact, when compared to traditional schemes, IoNCloud is able to reduce waste, saving up to 18% of resources.

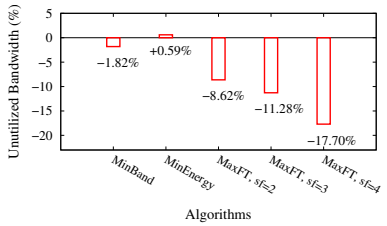


Fig. 6: Overall underutilization of resources.

IoNCloud can reduce resource underutilization, but it still suffers from some underutilization. As mentioned in the previous experiment, this happens because the current implementation of IoNCloud performs bandwidth deallocation at group granularity (as opposed to application granularity).

Ratio of Allocated VMs. This metric shows the proportion of VMs that were allocated in servers. *Higher values are better*, as the revenue of the cloud provider is proportional to the number of VMs it allocates. Figure 7 shows the difference between VM allocation ratios. As observed, IoNCloud performs better for all algorithms. Although the number of slots and VMs is the same, the allocation ratio differs depending on the allocation goal. This is because VMs can only be allocated if there is enough bandwidth for guaranteeing the setup of virtual links. Hence, reducing the amount of allocated bandwidth (as seen in Figure 4) increases the acceptance ratio of VMs in the cloud platform (since bandwidth is the bottleneck resource).

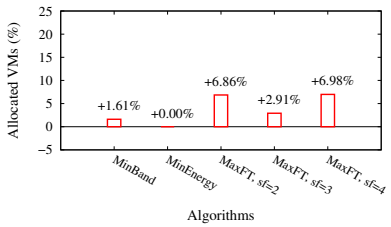


Fig. 7: Ratio of VMs that were placed in physical servers.

To understand the behavior of VM rejection, we perform experiments in a scenario were all datacenter links have unlimited bandwidth. Table I shows a comparison between both scenarios: normal and unlimited bandwidth. As can be observed, the assumption that bandwidth consumption interferes in VM allocation is verified, since all methods achieve 100% allocation with unlimited bandwidth. Note that *MinEnergy* is the only algorithm that achieves 100% VM allocation ratio under normal conditions. This is because VMs are packed together and fragmentation is minimal, thus, the majority of VMs will be closer. When minimizing bandwidth (*MinBand*), VMs may be allocated on free slots that are far from each other, which means that virtual links have a higher probability of reaching a bottlenecked physical link. *MaxFT* worsens this behavior, as it explicitly allocates VMs farther from each other.

TABLE I: VM allocation ratio with normal and unlimited bandwidth capacity on links.

VM placement goal	Bandwidth	
	Normal	Unlimited
MinBand	0.929	1
MinEnergy	1	1
MaxFT, sf=2	0.845	1
MaxFT, sf=3	0.892	1
MaxFT, sf=4	0.888	1

Link Sharing and Management Overhead. We also measure the number of reservations over each link in the datacenter network. Figure 8 shows the percentile difference of the maximum number of virtual links allocated in the network. We find that IoNCloud results in a significantly lower number of reservations to be managed (which can be as high as 22.32% less). In an environment as large and dynamic as a cloud platform, where network devices are limited in terms of the amount of control state and the rate at which these states can be updated, this typically results in a reduced reservation management overhead. Furthermore, during the experiments, we observed relatively small absolute values (an overall value of less than 10,000) for the number of reservations for all

[4]In our experiments, there were no reserved resources at the core.

strategies. This reflects the spatial locality applied by the allocation algorithms and suggests that the bandwidth reservation schemes can be accomplished using technologies already available in current datacenters (e.g., using rate-limiters in off-the-shelf switches or programmability of hypervisors [9]).

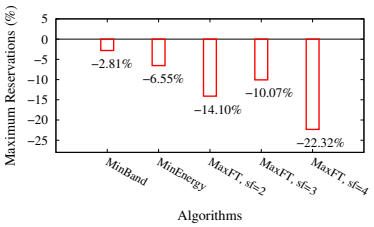


Fig. 8: Maximum number of allocated virtual links.

V. Design Discussion

Datacenter network topology. Current datacenters are typically implemented through a multi-rooted tree topology [11]. Therefore, in this paper, we focus on this kind of topology to show the benefits of IoNCloud. However, IoNCloud can be easily adapted for other topologies, such as random graphs [25]. In particular, it can be applied to multi-path topologies, both where load balancing is uniform across paths and where it is not uniform. For the first case (e.g., Fat-Tree), a single aggregate link can be used as a representation for a set of parallel links for bandwidth reservation [6], [8]. For the latter, IoNCloud would have to use an additional layer at hypervisor-level to control each path and its respective bandwidth for communication between VMs.

Online allocation of applications. IoNCloud allocates groups of applications in order to increase datacenter resource utilization. In this context, there is, at least, two ways of robustly providing online allocation for incoming application requests: i) by allocating an incoming application to an existing group; and ii) by allocating requests according to time slots. The first approach is straightforward, but may introduce some overhead to manage network resources when expanding an existing group. The second one (which we employed in our evaluation) takes advantage of high churn in cloud environments [4]. Thus, for each time slot (i.e., a predefined time period), IoNCloud can allocate the set of incoming requests by grouping them according to their bandwidth demands, without modifying previously allocated groups (less overhead).

Generality of the network model. Currently, IoNCloud adopts a single network model for all VMs of the same application. Nonetheless, it requires no modification when considering VMs of the same application with distinct network profiles. However, it may add some complexity to the resource allocation process. Another option is to extend the system to enforce per-VM traffic models by reserving bandwidth on links according to the VM with the highest demand in each application (at the cost of some underutilization).

VI. Conclusions and Future Work

We have introduced IoNCloud, a system that provides network predictability while minimizing resource underutilization and management overhead. To achieve this, IoNCloud groups applications in VNs according to their temporal bandwidth usage. Evaluation results show the benefits of our strategy, which is able to use available bandwidth more efficiently, reducing allocated bandwidth, network underutilization and management overhead. In future work, we intend to extend IoNCloud in two ways: i) by considering other objectives for application grouping; and ii) by adding VM migration to minimize network traffic.

Acknowledgements

This work has been supported by MCTI/CNPq/Universal (Project Phoenix, 460322/2014-1), MCTI/CNPq/CT-ENERG (Project ProSeG, 33/2013) and Microsoft Azure for Research grant award.

References

[1] F. R. Dogar et al., "Decentralized Task-Aware Scheduling for Data Center Networks," in *ACM SIGCOMM*, 2014.

[2] J. Lee et al., "Application-driven bandwidth guarantees in datacenters," in *ACM SIGCOMM*, 2014.

[3] J. Guo et al., "On Efficient Bandwidth Allocation for Traffic Variability in Datacenters," in *IEEE INFOCOM*, 2014.

[4] A. Shieh et al., "Sharing the data center network," in *USENIX NSDI*, 2011.

[5] D. Abts and B. Felderman, "A guided tour of data-center networking," *Comm. ACM*, vol. 55, no. 6, Jun. 2012.

[6] H. Ballani et al., "Towards predictable datacenter networks," in *ACM SIGCOMM*, 2011.

[7] R. Shea et al., "A Deep Investigation Into Network Performance in Virtual Machine Based Cloud Environment," in *IEEE INFOCOM*, 2014.

[8] L. Popa et al., "ElasticSwitch: Practical Work-Conserving Bandwidth Guarantees for Cloud Computing," in *ACM SIGCOMM*, 2013.

[9] D. Xie et al., "The only constant is change: Incorporating time-varying network reservations in data centers," in *ACM SIGCOMM*, 2012.

[10] M. Chowdhury et al., "Efficient Coflow Scheduling with Varys," in *ACM SIGCOMM*, 2014.

[11] H. Ballani et al., "Chatty tenants and the cloud network sharing problem," in *USENIX NSDI*, 2013.

[12] V. Jeyakumar et al., "EyeQ: practical network performance isolation at the edge," in *USENIX NSDI*, 2013.

[13] T. Benson et al., "Network traffic characteristics of data centers in the wild," in *ACM IMC*, 2010.

[14] X. Wang et al., "Carpo: Correlation-aware power optimization in data center networks," in *IEEE INFOCOM*, 2012.

[15] L. Chen and H. Shen, "Consolidating Complementary VMs with Spatial/Temporal-awareness in Cloud Datacenters," in *IEEE INFOCOM*, 2014.

[16] K. LaCurts et al., "Choreo: Network-aware task placement for cloud applications," in *ACM IMC*, 2013.

[17] C. Guo et al., "SecondNet: a data center network virtualization architecture with bandwidth guarantees," in *ACM CoNEXT*, 2010.

[18] H. Rodrigues et al., "Gatekeeper: supporting bandwidth guarantees for multi-tenant datacenter networks," in *USENIX WIOV*, 2011.

[19] L. Popa et al., "FairCloud: sharing the network in cloud computing," in *ACM SIGCOMM*, 2012.

[20] D. S. Marcon et al., "Trust-based grouping for cloud datacenters: improving security in shared infrastructures," in *IFIP Networking*, 2013.

[21] N. Chowdhury et al., "Virtual Network Embedding with Coordinated Node and Link Mapping," in *IEEE INFOCOM*, 2009.

[22] V. Mann et al., "VMFlow: leveraging VM mobility to reduce network power costs in data centers," in *IFIP Networking*, 2011.

[23] P. Bodík et al., "Surviving failures in bandwidth-constrained datacenters," in *ACM SIGCOMM*, 2012.

[24] S. Kandula et al., "The nature of data center traffic: measurements & analysis," in *ACM IMC*, 2009.

[25] A. Singla et al., "High Throughput Data Center Topology Design," in *USENIX NSDI*, 2014.