2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems

Managing Network Reservation for Tenants in Oversubscribed Clouds

Mayank Mishra
IIT Bombay, India
mayank@cse.iitb.ac.in

Partha Dutta
Xerox Research Center, India
partha.dutta@xerox.com

Praveen Kumar
IBM Research, India
praveek7@in.ibm.com

Vijay Mann
IBM Research, India
vijamann@in.ibm.com

***Abstract*—As businesses move their critical IT operations to multi-tenant cloud data centers, it is becoming increasingly important to provide network performance guarantees to individual tenants. Due to the impact of network congestion on the performance of many common cloud applications, recent work has focused on enabling network reservation for individual tenants. Current network reservation methods, however, do not gracefully degrade in the presence of network oversubscriptions that may frequently occur in a cloud environment.**

In this context, for a shared data center network, we introduce Network Satisfaction Ratio (NSR) as a measure of the satisfaction derived by a tenant from a given network reservation. NSR is defined as the ratio of the actual reserved bandwidth to the desired bandwidth of the tenant. Based on NSR, we present a novel network reservation mechanism that can admit time-varying tenant requests and can fairly distribute any degradation in the NSR among the tenants in presence of network oversubscription. We evaluate the proposed method using both synthetic network traffic trace and representative data center traffic trace generated by running a reduced data center job trace in a small testbed. The evaluation shows that our method adapts to changes in network reservations, and it provides significant and fair improvement in NSR when the data center network is oversubscribed.

I. Introduction

Motivation. Many cloud data centers support multiple tenants with diverse workloads. The workload of a tenant consists of one or more jobs, and the tenant's resource requirement consists of compute, memory, storage and network resources. These requirements generally vary with time, depending on the type and scale of individual jobs. For ensuring performance isolation among tenants, cloud operators have traditionally provided either dedicated compute, memory and storage resources, or reservations over slices of these resources, for each tenant. Since performance of many popular cloud applications, such as those based on map-reduce framework [1], depends significantly on network resources, some recent papers have proposed extension of tenant reservation to network resources [2], [3], [4], [5], [6], [7], [8].

Current network reservation methods, however, do not provide a mechanism to gracefully degrade the network reservation when the data center network is oversubscribed. Such a scenario may arise when new tenants join, or when tenants' network requirements increase. Network oversubscription may also occur if the network infrastructure is underprovisioned by design by the cloud providers, in order to keep CAPEX costs low in price-sensitive markets.

Network reservation methods that provide absolute bandwidth guarantees to the tenants may reject reservation requests if they cannot be satisfied over existing spare capacity in the network [3], [2], [4], [5]. Such a hard admission control is not only undesirable for the tenants, it may also indirectly result in lower average link utilization, and is therefore, deleterious for the cloud operator. For example, apprehending a rejection of future bandwidth increase requests, at the beginning of its tenancy, a tenant may request a bandwidth reservation that is close to the peak rate, resulting in a low average link utilization during most of its tenancy.

On the other hand, methods that propose fair sharing of network links among the tenants, characterize the requirements of a tenant (or a flow or a VM) using a weight. These methods, however, do not directly use the desired bandwidths of the tenants [6], [7]. In such a system, the bandwidth share for a tenant may arbitrarily degrade if there is a significant increase in the number of tenants or their network requirements. On a related note, it is important to observe that the fair sharing of link bandwidth among TCP flows is inadequate for tenants' network reservations because TCP's fair sharing of bandwidth depends on the network characteristics of the flows, such as loss rate and RTT, but it is oblivious to tenants' desired bandwidths.

Contribution. In this work, we present a network reservation mechanism for cloud tenants that provides a balance between absolute bandwidth guarantees and relative bandwidth sharing approaches, and therefore, gradually and fairly reduces the network reservation during oversubscriptions. We first introduce *Network Satisfaction Ratio (NSR)* of a tenant as measure of the satisfaction a tenant derives from a network reservation. At a high level, we define NSR of a tenant as the ratio of its reserved bandwidth to the desired bandwidth. Since this work is a first step towards managing satisfaction derived by tenants from network reservation in cloud, we use a simple definition of tenant's network satisfaction. More sophisticated measures of satisfaction can be obtained by considering real-time network performance metrics, or other functions of reserved and desired bandwidths.

Using NSR, we present a method that periodically performs network resource allocation with the aim of fairly satisfying the tenants. In particular, if network reservation requests of every tenant cannot be fully satisfied, the requests and existing reservations are reduced such that the minimum NSR over all tenants is maximized. Following [3], [9], we consider the abstraction of a virtual cluster (also known as Hose model) for specifying the network reservation of a tenant. After the initial placement of the tenant, the method uses flow rerouting and a limited number of VM migrations to handle subsequent bandwidth reservation requests. The method is executed periodically, so as to handle changes in tenants' network

Mayank Mishra and Partha Dutta did this work at IBM Research, India

1526-7539/13 $26.00 © 2013 IEEE
DOI 10.1109/MASCOTS.2013.13

requirements as well as arrival and departures of tenants. We use a novel observation about the optimal bandwidth requirement of a virtual cluster (VC) reservation to reduce the number of rerouting and migrations. Finally, to avoid arbitrary degradation in NSR in case of a large number of tenants or bandwidth requests, a request is admitted only if the minimum NSR over all tenants can be maintained above a certain predefined threshold.

We evaluate our method using trace-driven simulations. We use two sets of network reservation request traces. The first request trace is derived from the network traffic inside a small cluster when a reduced job trace from Facebook [10] is executed over the cluster. The second set is synthetically generated assuming certain time-varying tenant network traffic. As compared to the method in [3], our scheme shows significant and fair improvement in tenants' network satisfaction for both sets of traces, and adapts to changes in bandwidth requests, while maintaining same or better link utilization.

Roadmap. The rest of the paper is organized as follows. In Section II, we introduce NSR for a tenant, and recall the virtual cluster abstraction for network reservation. Section III presents an observation on the optimality of a virtual cluster reservation, which is used in our network reservation algorithm presented in Section IV. In Section V, we describe our method for collecting representative data center network traffic traces on a small testbed based on real data center job traces, and some observations on that data. We present our evaluation results in Section VI, using both representative data center traffic traces and synthetic traffic traces. Section VIII describes the related work, and Section IX concludes the paper with some directions for future work.

II. Network Satisfaction Ratio of a Tenant

We consider a cloud data center where the bandwidth request of every tenant has equal priority. In an oversubscribed scenario, ideally we would like to gracefully and fairly degrade the quality of the network reservation of all tenants. Towards this end, we introduce *Network Satisfaction Ratio (NSR)* as a measure of the satisfaction that a tenant derives from a given reserved bandwidth, with respect to the tenant's desired bandwidth. We note that, in the case where tenants have different priority levels, NSR of a tenant can be redefined to include a multiplicative weight that reflects the tenant's priority level. Next, we describe a representation of a tenant's network requirement using virtual network and virtual cluster [3], and then introduce our definition of NSR.

The physical network topology of the data center is modeled using a graph where physical switches and physical servers are denoted by nodes, and physical links by edges. This work considers two kinds of resources for reservation in a data center: the network bandwidth of the links in the data center, and the number of slots at a physical server, where each slot can host one VM.[1]

Both the desired bandwidths and the reserved bandwidths among the VMs of a tenant are specified using *virtual networks*. Nodes in a virtual network represent either the VMs of the tenant, or *virtual switches*. Virtual switches in a virtual network are used to describe more general bandwidth requirements that cannot be expressed as a direct bandwidth requirement between a pair of VMs. The weight of an edge (also called *virtual link*) in a virtual network is the bandwidth specified between its end nodes. In particular, when a virtual network is provisioned on a physical network, a VM in the virtual network is mapped to a VM slot at a physical server, but a virtual switch may or may not be mapped to a physical node (switch or server).

A common virtual network is a clique (among the VMs), which specifies a bandwidth between every pair of VMs of a tenant. In this work, we consider a virtual network of the form *virtual cluster* (VC) [3], which has earlier been studied as Hose Model for VPNs [9]. In a virtual cluster, there is an intermediate virtual switch (VS), and a (sum of incoming and outgoing) bandwidth is specified between every VM and the VS, as shown in Figure 1. Note that, each VM may have a different bandwidth requirement to the VS.

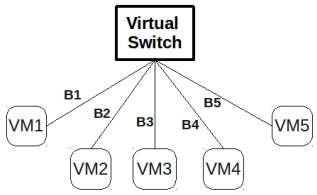


Fig. 1: A Virtual Cluster.

A network reservation method finds an appropriate resource reservation for a tenant, and modifies the reservation based on subsequent tenant requests. The reservation method should perform resource allocation (either for an initial placement or subsequent modification) so as to maximize the tenant's satisfaction from the reservation. We characterize the satisfaction level of a specific network resource reservation for a tenant through Network Satisfaction Ratio (NSR). The NSR of a reserved virtual network, given a desired virtual network (over the same physical network topology), is defined as the minimum ratio of the reserved bandwidth to the desired bandwidth, over all virtual links. The NSR of a tenant at a given point in time, is the NSR of its current reserved virtual network given its last specified desired virtual network.

Note that NSR characterizes only the quality of network provisioning, i.e., the satisfaction level of a tenant for a given network provisioning with respect to its requested network provisioning, and therefore, depends only on the reserved and desired bandwidths. NSR is not a real-time QoS metric, and does not consider the actual tenant traffic.

III. Virtual Cluster and Virtual Switch Placement

In this section, we consider the VC reservation for a single tenant. Consider a desired VC, vc, where $br_{vc}(i)$ denotes the bandwidth requirement of VM i to the virtual switch of vc. A bandwidth reservation R for vc, given a data center network topology G and placement for all VMs to physical server, is specified as follows: R specifies bandwidth $R(e)$ (possibly 0) reserved on each edge e of G such that, any traffic requirement of the tenant, in which the total traffic (sum of incoming and outgoing) for each VM i is at most $br_{vc}(i)$, can be satisfied by routing using the reserved bandwidths.

[1] Since allocation of network resources is the focus of this paper, we ignore the variation in the compute requirements across VMs, as well as ignore the storage and the memory requirements.

Optimal bandwidth reservation. Consider a tree data center network topology T, where the root is a core router, the internal nodes are the switches, and the leaves are the physical servers.[2] One or more VMs can be placed on a leaf node. For ease of presentation, we assume that edges and nodes in T have a natural top to bottom ordering, with the root at the top and leaves at the bottom. Assume that all the VMs of the tenant have been placed on physical servers.

For a leaf (i.e., a server) v, let $b(v)$ denote the total desired bandwidth of all VMs that are placed on v (according to the bandwidth requirement of those VMs in vc). Upon removing an edge e from T, T is divided into two subtrees, namely the lower subgraph $G_l(e)$ (i.e., the subtree rooted at the lower of the two end-nodes of e) and the remaining upper subgraph $G_u(e)$ (i.e., the subgraph obtained by removing $G_l(e)$ and e from T). Let $l(e)$ ($u(e)$) denote the set of VMs in the lower (resp., upper) subgraph, and let $B_l(e)$ (resp., $B_u(e)$) denote total desired bandwidth of those VMs. Figure 2 gives an example of terms defined above.

For a non-root node w of T, we similarly define the lower subgraph $G_l(w)$ as the subtree rooted at w that contains all descendants of w, and the upper subgraph $G_u(w)$ as the subgraph obtained by removing $G_l(w)$ and the edge connecting w to its parent node. We similarly define $l(w)$, $u(w)$, $B_l(w)$ and $B_u(w)$.

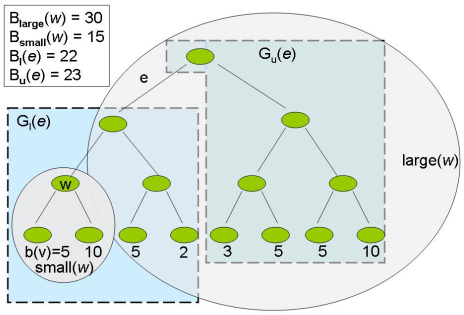


Fig. 2: An example with node w is in $G_l(e)$ (Lemma 2).

We now recall a result from [3] on the optimal reservation for VC when the network topology is a tree. This result and its basic correctness argument is informally presented in [3]. Proff of Lemma 1 is omitted due to lack of space.

Lemma 1. *(From [3]) To reserve bandwidth for a virtual cluster vc over T, it is necessary and sufficient to reserve a bandwidth of Min$\{B_l(e), B_u(e)\}$ on each edge e of T.*

We note that Min$\{B_l(e), B_u(e)\}$ is the maximum possible traffic on the edge e when the traffic pattern follows the specification of virtual cluster vc. In other words, the actual traffic on e when the traffic pattern follows vc can vary between 0 and Min$\{B_l(e), B_u(e)\}$.

Virtual Switch placement. Next, we present a generalization of the above lemma, which serves as the basis for our algorithm. Consider a non-root node w in tree topology T, and VM placements for virtual cluster vc. Let $large(w)$ be $G_u(w)$ if $B_u(w) \geq B_l(w)$, and $G_l(w)$, otherwise. Let $small(w)$ be the the subgraph ($G_u(w)$ or $G_l(w)$) that is different form $large(w)$. Let $B_{large}(w)$ ($B_{small}(w)$) be the sum of the desired bandwidth of all the VMs in $large(w)$ (resp., $small(w)$). Figure 2 provides an example for the above definitions.

For a non-root node w with $B_{small}(w) > 0$, we define the *bandwidth offset ratio* $o_{vc}(w)$ of w as $\frac{B_{large}(w)}{B_{small}(w)}$. Note that the minimum value of $o_{vc}(w)$ is 1. Next, for any node w in the topology, we define a *w-virtual switch reservation* ($vs_{vc}(w)$-reservation) as the reservation in which, for each VM i, a bandwidth of $br_{vc}(i)$ is reserved on every link on a shortest path from i to w in the topology. We now show that for a tree topology, a $vs(w)$-reservation is within a $o_{vc}(w)$ factor of the optimal, where the optimal is given by Lemma 1. In the remainder of the discussion, we omit the virtual cluster vc subscript when it is obvious from the context.

Lemma 2. *For any non-root node w in T with $B_{small}(w) > 0$, if $vs(w)$-reservation is done over T for a virtual cluster vc, the actual reservation for each edge is within $o(w)$ times the optimal.*

Proof: Consider an edge e in T. Without loss of generality, assume that $B_l(e) \leq B_u(e)$. Then, from Lemma 1, the optimal reservation on e is $B_l(e)$. We consider two cases.

Node w is in $G_u(e)$. In this case, in $vs(w)$-reservation, each VM i in $l(e)$ reserves a bandwidth of $br(i)$ on the (shortest) path from i to w. All these paths include e, and these are the only reservations on e. Thus, the actual $vs(w)$-reservation on e is $B_l(e)$, which is also the optimal reservation, as mentioned above. Since $o(w)$ is at least 1, the actual reservation for e is within $o(w)$ times the optimal.

Node w is in $G_l(e)$. In this case, in $vs(w)$-reservation, each VM i in $G_u(e)$ reserves a bandwidth of $br(i)$ on the (shortest) path from i to w. All these paths include e, and these are the only reservations on e. Thus, the actual $vs(w)$-reservation on e is $B_u(e)$. See Figure 2.

Consider the two subgraphs derived from w, $large(w)$ and $small(w)$. Since w is in $G_l(e)$, one of the subgraphs derived from w is a subgraph of $G_l(e)$, and $G_u(e)$ is a subgraph of the other subgraph derived from w. Since, $B_u(e)$ is higher than $B_l(e)$, $G_u(e)$ is a subgraph of $large(w)$, and $small(w)$ is a subgraph of $G_l(e)$. Thus, $B_u(e) \leq B_{large}(w)$, and $B_l(e) \geq B_{small}(w)$. Consequently, the ratio of the actual bandwidth reserved on e, which as discussed above is $B_u(e)$, to the optimal bandwidth reserved on e, which is $B_l(e)$, is at most $\frac{B_{large}(w)}{B_{small}(w)} = o(w)$. ■

Before concluding this section, we briefly state the two special cases in VS placement at a node w and its associated w-virtual switch reservation. First, when $B_{small}(w) = 0$, the ratio of the reserved bandwidth to the optimal bandwidth can be arbitrarily large. Nodes with $B_{small}(w) = 0$ are trivial to identify, and should not be considered for VS placement. Second, when the root node is selected as VS w, there are two cases. If for every child x of w, $B_l(x) \leq B_u(x)$, then the reserved bandwidth is the optimal bandwidth for every edge. Otherwise, if for any child x, $B_l(x) > B_u(x)$, then let y be

[2] Although the observations in this section are for a tree data center topology, the algorithm we present in the next section can handle non-tree topologies.

the child of w with maximum value of $\frac{B_l(y)}{B_u(y)}$. Then the ratio of the reserved bandwidth to the optimal bandwidth for every edge is at most $\frac{B_l(y)}{B_u(y)}$.

Observations for managing tenant reservations. Given a routing tree, Lemma 2 points out that nodes with low bandwidth offset ratio should be preferred for VS placement. This is because, from Lemma 2, a low bandwidth offset ratio results in a corresponding bandwidth reservation on each edge that is close to the optimal reservation. Note that, even if a VS placement on a node results in a low bandwidth offset, the corresponding routing may not feasible due to insufficient spare capacity on the edges. Therefore, multiple nodes may need to be considered for VS placement.

If the data center topology is not a tree, then a routing tree T' should be selected (among the candidate subtrees of the topology) such that some VS placement over T' gives a low bandwidth offset ratio. These two observations are at the heart of the reservation management algorithm presented in the next section.

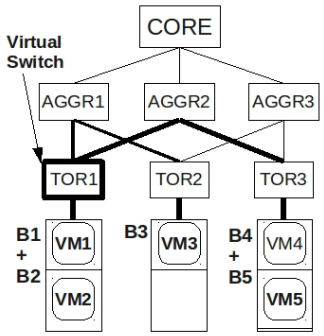


Fig. 3: VC placed on the data center. Thicker lines show the demand routes.

IV. Algorithm

Our algorithm for managing NSR of a tenant consists of (1) initial VM placement and flow routing when the virtual cluster is created, and (2) subsequent rerouting of flows and limited number of VM migrations when the desired bandwidths in the virtual cluster is changed by the tenant. We now describe these two steps in more detail. The objective of our algorithm is to maintain as high an NSR as possible for all tenants (by trying to maximize the minimum NSR among tenants), while performing limited or no (expensive) VM migrations. Note that the algorithm uses the observations from Section III from VS placement and demand routing, and the algorithm can be applied to a tree as well as a non-tree data center topology.

A. Initial Placement

When the system receives a request for a desired virtual cluster (VC) of a tenant for the first time, an initial placement scheme is used for placing the VMs of the VC, and for reserving link bandwidth for routing the flows of the VC. In this work, we use an initial placement scheme that is an extension of the VC placement scheme in [3]. We note that the VC placement algorithm in [3] does not handle subsequent changes in the desired bandwidths of VMs, which is the main contribution of our algorithm.

Bandwidth aware VC placement. This method is an extension of VC placement algorithm in [3]. The algorithm, while taking into account the desired bandwidth and the number of VMs, places the VC on the lowest subtree that can accommodate the VMs and the flows.[3] We extend the algorithm to handle heterogeneous VM bandwidths and non-tree topologies. Although the algorithm tries to maximize the minimum NSR among the tenants, we maintain a minimum NSR threshold α. We assume that if a desired VC cannot be provisioned with NSR of at least α (i.e., all VMs of the VC with bandwidths reduced by a factor of α), then the data center network is severely overloaded, and hence, we do not accept new reservation requests. Parameter α is selected by cloud provider and is the same for all tenants.

To address heterogeneous VM bandwidths within a VC we do the following steps. While considering a sub-tree for placing the VC or a subset of nodes of VC: in addition to taking into account the number of VM slots of the subtree, we also consider the number of bandwidth slots. Here the bandwidth slot of a subtree is defined as $\lfloor \frac{avl\_bw}{\alpha.avg\_bw} \rfloor$, where avl_bw is the available bandwidth of the link from the root of subtree to its parent, and avg_bw is the average bandwidth of all the VMs in the VC. The minimum of the two capacities (bandwidth slots and VM slots) decides the number of VMs that can be placed on the subtree. If all VMs of the VC with bandwidths reduced by a factor of α can be placed, the VC is accepted for placement, otherwise, it is rejected. In addition, to maximize NSR, the flow reservation for the VC over each link is set to the maximum possible value (based on the available link capacity) which is between α and 1 times the optimal reservation value given by Lemma 1.

In a non-tree topology, there can be multiple subtrees with a selected root node and the set of leaf nodes. The number of such possible subtrees depends on the degree of redundancy in the network links and the number of levels in the subtree itself. As the number of such subtrees can become very large, we only consider a fixed maximum number of such subtrees and select the one which gives the highest NSR with value above the minimum of α.

Note that, in the above scheme, a VC is placed over a subtree of the physical network topology: the VMs are placed on the VM slots of the hosts in the subtree, and the bandwidth is reserved over the links in the subtree. We call this subtree the routing tree of the VC placement. After selecting an initial VC placement, we select an initial virtual switch (VS) among the nodes in the routing tree. VS is selected to be the node that has the lowest bandwidth offset ratio as described in Section III. See Figure 3.

B. Changing VC reservation

A tenant may request to change the bandwidth reservation of the flows of VMs or may even add or remove VMs from a VC. For simplicity of presentation, we do not consider addition or removal of VMs from VC in this work.

For describing the algorithm we first define a specific kind of reservation. For a given routing tree T, a virtual cluster

[3] In [3], the link bandwidth requirements are calculated using Lemma 1 that we mentioned earlier, and the network topology is assumed to be a tree.

vc and VS placement on node w, we define $vs_{vc}(w, \beta)$-reservation as a $vs_{vc}(w)$-reservation in which the reservation from each VM to VS is β times its desired bandwidth, where $\beta \leq 1$ (see Section III for a definition of $vs_{vc}(w)$-reservation). In the algorithm, the value of β is varied from 0 to 1, so as to find out a reservation that is closest to the desired bandwidths. Note that, although the underlying data center topology can be a non-tree, the algorithm selects routing subgraphs (i.e., subgraphs of the topology over which demand routing is performed) that are trees.

Periodically, the algorithm considers the pending requested changes to the VC. The requested changes in bandwidth reservation of the demands of VMs are considered in the descending order of the desired bandwidths. Since, higher desired bandwidth for a VM requires higher bandwidth reservation for maintaining the same value of NSR, the algorithm gives priority to the VMs with high desired bandwidths. Following steps are taken for each bandwidth request b of a VM with requested bandwidth rate r_{new} and earlier reserved bandwidth rate r_{old}.

- Let the available bandwidth on the current route be avl. If current route of the VM demand is able to satisfy α times the desired bandwidth, i.e., $avl \geq r_{new} * \alpha$, then the current route is used. The new reserved bandwidth is set to $min(r_{new}, avl)$. If $avl < r_{new} * \alpha$, then we try the next step.
- In this step, the VM demand is rerouted along a new route which can support at least bandwidth of $r_{new} * \alpha$. Please note that there can be multiple routes which can satisfy the requested rate. Our algorithm tries to find the shortest such route. If such a route cannot be found, we move to the last step.
- The bandwidth reservation request is rejected and VM demand continues to utilize r_{old}.

It can be seen that the requested increase in the bandwidth reservation may not always get fulfilled due to unavailability of bandwidth on the physical links. As a result, NSR of the VC may decrease over time. This is because, for each tenant VC, its NSR is computed by comparing the reserved bandwidths with the last desired bandwidths of the VC. On the other hand, the NSR of a VC may also increase if its desired bandwidth decreases. We therefore use a periodic NSR improvement method which is described next.

C. Improving NSR

Periodically, the algorithm tries to improve the NSR of the placed VCs which have NSR below the predefined tolerance level α. The VCs are considered one at a time in the ascending order of NSR. The algorithm considers VCs in ascending order of their NSR for improvement because the overall objective is to maximize the minimum NSR among the VCs. For each selected VC vc, the algorithm attempts to do flow rerouting and VM migrations to improve the NSR. First, we select the current routing tree for the VC. For each node w in this tree, we compute the bandwidth offset ratio if w is selected as the VS, and $vs_{vc}(w)$-reservation is done. Guided by Lemma 2, next we select a set of nodes, whose offset ratio is above a pre-defined threshold, as potential candidates for VS, and we consider $vs_{vc}(w)$-reservation for each candidate w. Among all candidates for VS, we select the node w which maximizes the NSR of VC, i.e., w has the highest value of β such that $vs_{vc}(w, \beta)$- reservation can be performed over the residual link capacities of the current routing tree for the VC. If β is equal to or higher than the minimum tolerance level of NSR, α, then we reroute the flows and change the link reservation according to $vs_{vc}(w, \beta)$-reservation (and we say that VS is migrated to node w). Otherwise, additionally, we consider migrating the VMs of the VC, as described below.

For VM migration, we consider the VMs of the VC in the ascending order of the ratio between current reserved bandwidth and the desired bandwidth. A VM is migrated to the closest physical server s (from its current server) such that s has a free VM slot and there a path from s to w (VS selected in the previous step that maximizes NSR) which has enough residual bandwidth to support the desired bandwidth of the VM (Figure 4). After each VM migration, the NSR of the VC is recomputed until the threshold on the number of concurrent VM migration is exceeded or the NSR crosses the tolerance threshold.

The time interval between two consecutive invocations of the NSR improvement method provides a tradeoff between faster admission of tenants' bandwidth change requests (when the interval is short) and low overhead of running the method and its subsequent rerouting and VM migration (when the interval is long). The duration of the interval should be fine-tuned for each data center based on its size, cost of VM migration and rerouting, and how frequently the tenant requests change. In our simulations, we consider an interval of 1200 seconds.

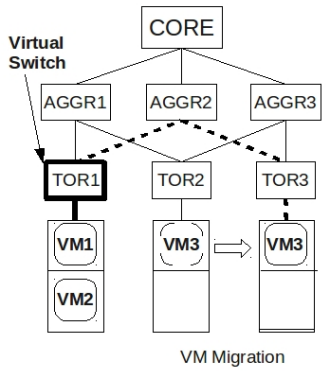


Fig. 4: VM3 getting migrated. Dotted line shows the new route of VM demand.

V. Representative Data Center Traffic Traces

A. Testbed details

To evaluate our algorithm, we use network traces obtained from a testbed running different MapReduce jobs. We use a 60-node cluster running Hadoop 1.1.1 [11] to perform our experiments. The 60 virtual machines are distributed across 6 physical hosts and each physical server hosts 10 nodes. All the physical hosts are IBM x3650 M2 servers with Intel Xeon CPU E5675 processor with a total of 12 cores on each server. All the hosts and the VMs run 64-bit Ubuntu 12.04.1 server with Linux 3.2.0-29-generic kernel. We use KVM to run the VMs and assign 2GB of RAM and one core to each VM. Every physical host has an OpenVSwitch (OVS) running and all the

nodes placed on the host are connected to it. The physical interfaces bridged with the OVS on the hosts are connected to a single physical switch using 10 Gbps links.

B. Trace generation

We generate a workload on our cluster that is representative of the workload in a real data center. For this, we deploy the Statistical Workload Injector (SWIM) for MapReduce, from https://github.com/SWIMProject-UCB/SWIM/wiki [10], [12] on our cluster. SWIM provides a set of MapReduce workloads from real production systems and a tool to generate representative test workloads by sampling historical MapReduce cluster traces which can be used to replay the representative workload on other clusters. SWIM provides representative workload reproduced from the original trace that maintains the distribution of input, shuffle, and output data sizes, and the mix of job submission rates, job sequences and job types [12]. Using SWIM, we pre-populate the HDFS (Hadoop Distributed File System) on our cluster using synthetic data, scaled to the 60-node cluster, and replay the workload using synthetic MapReduce jobs. Once the replay is started, the jobs keep getting submitted to Hadoop at varying time intervals as calculated by SWIM and are handled by the default Hadoop scheduler. For our evaluation, we use the traces available in the SWIM workload repository which are based on Facebook's production data center. To measure the bandwidth utilization during the process, we enable NetFlow on the OVS switches with an active timeout of 1 second and monitor the traffic using our own NetFlow collector. Based on the production data center traces, we generate 18 different workloads which run for about 4 hours each. Thus, we collect network traffic data for a total of 72 hours. To get the actual bandwidth requirement of the nodes while executing a workload, the data rate should not be capped by the link capacity. Therefore, we ensure that the link capacity in the testbed is well above the required level for all our experiments. We use the data collected using this methodology for evaluating our algorithms through simulation and present our results in Section VI-A.

C. Observations

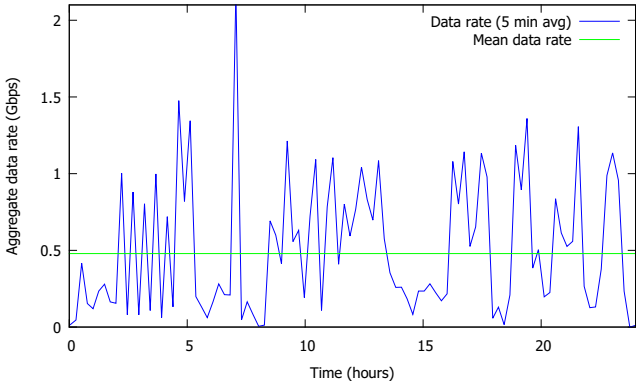


Fig. 5: Aggregate data transfer rate for a period of 24 hours

In addition to evaluating our algorithm, we also analyzed the network traffic data and observed a few interesting points. Figure 5 shows the variation of the aggregate data transfer rate (across all 60 nodes in the cluster) with time for a duration of 24 hours. The rate is averaged over 5 minutes to even out very short bursts. This data clearly shows that the bandwidth requirement for a tenant in a real data center is time-varying. We see that there are significantly long intervals when the bandwidth requirement is well above the mean value. The peak-to-mean ratio for the 5-min average bandwidth requirement is 4.6. 40% of the time, the requirement is above mean and 16% of the time, the requirement is more than twice the mean value.

Figure 6 shows the amount of data transferred between server pairs for 3 intervals of 15 minutes each. It is a scatter plot for the matrix containing the $\log$ of bytes sent from one node to another. The three subfigures represent the data for 3 consecutive 15-minute intervals. Since the samples were taken when the network load was varying (corresponding to around 3 hour mark in Figure 5), we see the difference in the three plots. Figure 6 (b) corresponds to a peak in Figure 5 while the other two correspond to the adjacent troughs. We notice that for each of the plots, the darkness is not uniform, implying that some of the nodes demand higher bandwidths. This serves as a motivation for a network reservation abstraction, such as a VC, with heterogeneous VM bandwidths. Further, the location of the darker points in the plot keep changing over time. This means that the set of nodes which require higher bandwidth keep changing with time. This corroborates one of the basic assumptions of this work that the bandwidth requirement of a tenant's network reservation should be allowed to change over time.

VI. Results

In this section we present the results of simulation based comparison of our method with other NSR managing schemes. For simulations, we have used both real world as well as synthetic data traces. Values of some parameters used in simulation are mentioned below.

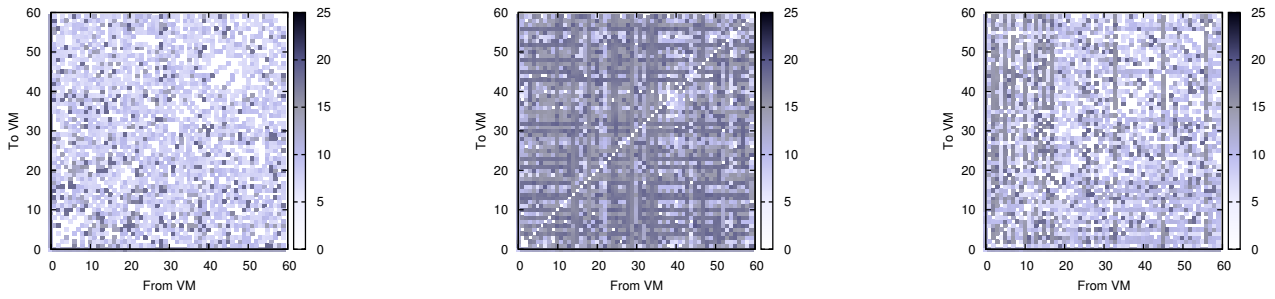
- Both α and β are set to 0.7.
- Number of candidate subtrees considered for VC placement are limited to a maximum of 5.

A. Simulation with Representative DC traffic

Network workload considered: We consider a MapReduce network traffic workload consisting of 18 tenants with 60 VMs per tenant. We assign each of the tenants a different trace that we generated using SWIM in section V. Each trace is a mix of various jobs with varying requirements. Thus, the set of 18 tenants running simultaneously represents a good multi-tenant data center workload.

Data center topology: We consider a synthetic CLOS topology with 300 servers where each server is capable of hosting 4 VMs and each link has the bandwidth capacity of 768Mbps. This capacity was selected to simulate an oversubscribed data center scenario, given that our data center traffic data over 60 VMs (per Tenant) has much lower traffic than a commercial data center. When we ran the simulation using the same-sized data center with 1Gbps links, we found that the data center is no longer oversubscribed and the NSR of all the tenants reached 1. It should be noted that the topology is a non-tree topology.

We have used the placement scheme described in Section IV-A for the initial placement of the VMs of the tenants. We



(a) 110 to 125 min (b) 125 to 140 min (c) 140 to 155 min

Fig. 6: Data transfer ($\log(Bytes)$) between server pairs for 15 minutes

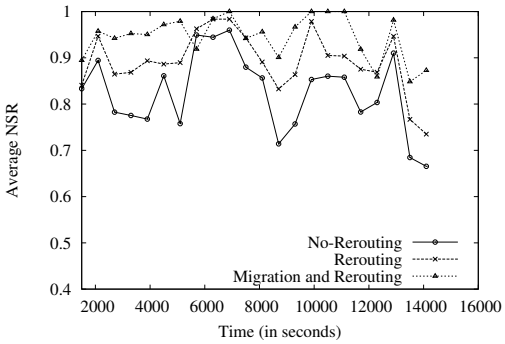


Fig. 7: Representative DC trace: Average of NSRs of tenants

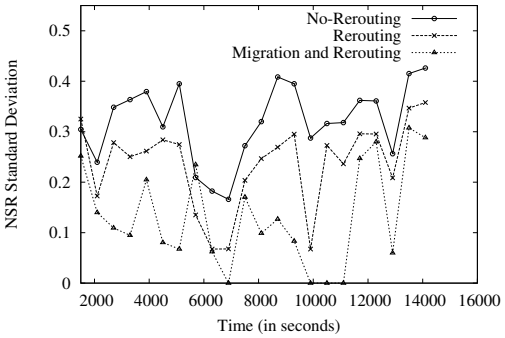


Fig. 8: Representative DC trace: Standard deviation of NSRs of tenants

compare three NSR maintaining approaches (including ours) mentioned below.

- No-Rerouting : Once the VC is placed, there is no change in the locations of the VMs and also no rerouting of the bandwidth reservations of the VMs.
- Rerouting : Once the VC is placed, there is no change in the locations of VMs and VS. Only demand rerouting is allowed when the current bandwidth reservation route is unable to support its increased bandwidth requirement.
- Migrations and Rerouting (Our Scheme): This scheme is discussed in detail in Section IV.

We now list the key results and observations from this simulation.

1. Figure 7 shows that our scheme provides better average NSR to the tenants than other schemes. Its performance is between 6% to 28% higher than the No-rerouting scheme and close to 10% higher than the Rerouting scheme. Our scheme occasionally provides the maximum possible NSR of 1 while other schemes are never able to achieve that level of NSR. The total number of VM and VS migrations which occurred during the simulations for all 18 tenants are 6 each.

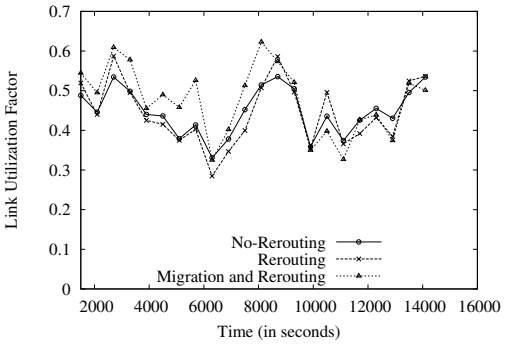


Fig. 9: Representative DC trace: Average utilization of links between ToRs and Aggregate Switches

2. Figure 8 shows that our scheme is much more fair to tenants in terms of bandwidth allocations as shown by the standard deviation of NSR across the tenants. Occasionally, for our scheme, the standard deviation of NSR becomes zero, which implies an equal NSR for all.

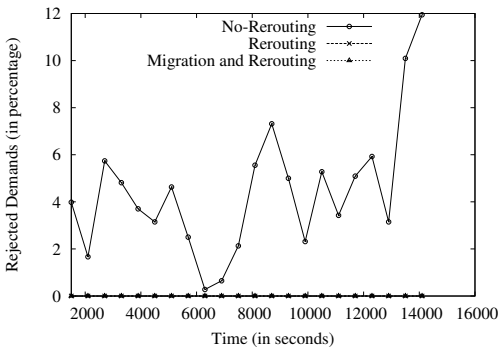


Fig. 10: Representative DC trace: Rejected demands as percentage of total number of requested demands

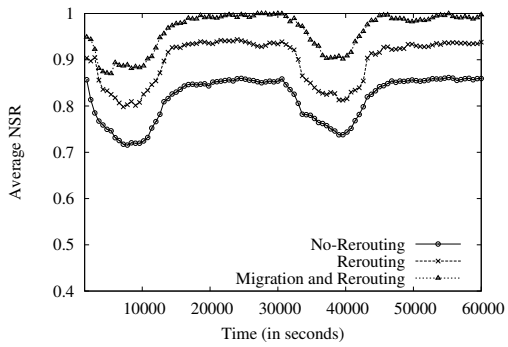


Fig. 11: Synthetic traffic trace: Average of NSRs of tenants

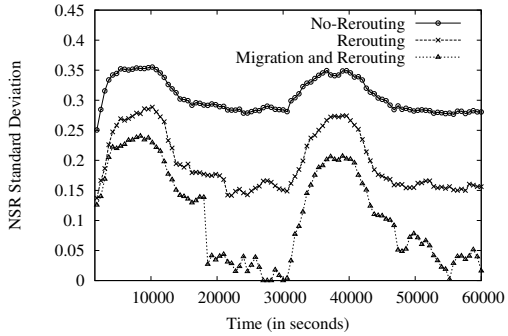


Fig. 12: Synthetic traffic trace: Standard deviation of NSRs of tenants

3. The average NSR provided by our method is less time-varying as compared to the others. This shows that our scheme is able to adapt to the varying network load.

4. We observed that the server to ToR links have lower average link utilization as compared to ToR to aggregation switch links, which in turn has lower link utilization as compared to aggregation to core switch links. Our scheme maintains similar link utilization as others while maintaining higher NSR (Figure 9).

5. Figure 10 shows percentage of demands which are rejected. A new demand is said to be rejected if there is no route in the data center between the demand end points which can fully support the demand. An already placed demand is said to be rejected if increase in its bandwidth requirement can not be supported. The rerouting module in our scheme helps to quickly adapt to the dynamically changing bandwidth demands, and thus, reduce the bandwidth reservation rejections.

B. Simulation with Synthetic Data traffic

In this section, we present the simulation results with synthetic network traffic traces with more tenants and using a much bigger data center.

Network workload: We consider a workload consisting of 80 tenants with an average of 40 VMs each. The bandwidth requirement of a VM is 50 Mbps on an average. The VMs vary their bandwidth reservation requirements with a standard deviation of 20Mbps and following a normal distribution. To smoothen out the traffic we use exponential smoothing with smoothing factor of 0.25. To mimic the dynamic nature of network load we vary the bandwidth required by the VMs alternately to high load and low load conditions. When the load is high, the average bandwidth requirement of VM is twice than that in low load scenario, i.e., 100 Mbps per VM. The increase and decrease in loads between the low and high periods are gradual, each taking 10% time of the total high load duration.

Data center topology: We consider a synthetic CLOS topology data center with 1024 servers with each server capable of hosting 4 VMs and links connecting different nodes have 1Gbps capacity.

We now list the key results and observations from this simulation.

1. Figure 11 shows that our scheme provides much better average NSR to the tenants. Its performance is between 20% to 26% higher than the No-rerouting scheme and between 10% to 16% higher than the Rerouting scheme. Also note that our scheme provides an average NSR close to 1 for a considerable amount of time whereas, other schemes are never able to achieve that level of NSR.

2. The standard deviation of NSR among the tenants achieved by our scheme is always lower than the other schemes, as shown in Figure 12. Occasionally it reaches zero and remains very close to zero when the data center is in low utilization phase. This shows that our scheme is fair across the tenants.

3. Our scheme utilizes the links marginally better than other schemes. This is evident from the Figure 13. This, coupled with zero demand rejection rate as shown in Figure 14, are essential to have higher NSR and efficient data center operations.

C. Adaptive placement

In our simulations, the initial placement of tenant VC plays a key role in deciding the NSR observed, especially for the unadaptive network reservation management schemes. This was evident when we experimented with simpler initial placement schemes, such as simple left to right packing of VMs on the physical servers. In that case, we found that the difference in performance between our scheme and other

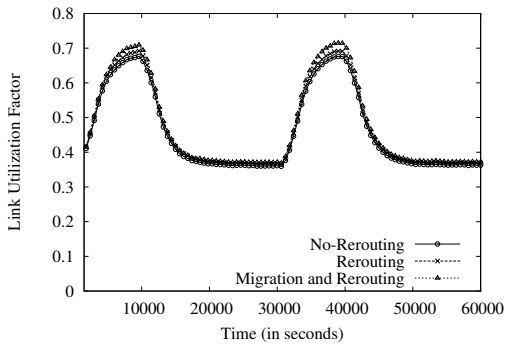


Fig. 13: Synthetic traffic trace: Average utilization of links between ToRs and aggregate switches

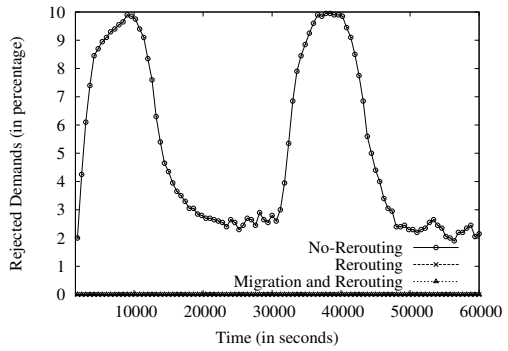


Fig. 14: Synthetic traffic trace: Rejected demands as percentage of total number of requested demands

schemes was much higher. Due to its adaptive nature, our scheme, after some initial period of low NSR, was able to achieve similar performance with simple initial placement scheme as the performance shown in above simulation results. Thus, we note that even if the initial placement is not done in the best possible way, an adaptive scheme like the one proposed in this work will make the placement better over time.

VII. Overheads involved in our mechanism

In this section, we discuss the overheads of the proposed NSR improvement mechanism. There are three activities that cause the overheads, namely, VM migration, VM flow rerouting and the overhead of running the algorithm itself.

VM Migration: The proposed mechanism tries to keep the number of VM migrations to a minimum by using flow rerouting whenever possible. VM migration is used only if rerouting of the flow is not sufficient to improve a tenant's NSR. The number of VM migrations is kept low because of its high cost [13], which involves transferring the VM image from the current physical server to the destination server. In the experiment mentioned in section VI-A with real data traces, we found that our scheme required only 6 VM migrations during 4 hours of simulation with 1080 VMs involved.

Rerouting: Rerouting of a VM flow needs much lesser effort than migrating the VM. In an OpenFlow network controller based network, it only involves configuration of the new forwarding rules (along with removing the old rules) on the switches. The installation of new rules on a switch requires only a few milliseconds. A good discussion about the overhead involved in configuration of forwarding rules on OpenFlow enabled switches is given in [14].

Cost of running our algorithm: As discussed in Section IV-C, our algorithm tries to improve the NSRs of tenants periodically. The calculations involved consist of finding new paths for rerouting the flows and finding new destination physical servers for migrating the VMs. We have implemented these routines in the form of a module (in Java) which currently works on top of a data center simulation [15]. The time taken for simulation consists of the time taken by the module to make the rerouting and migration decisions along with the simulation of a data center. For the 4 hour long "real data traffic" trace simulation (Section VI-A) consisting of 1080 VMs and a data center with 300 servers, the time taken per simulation run was close to 180 seconds. The NSR improvement was invoked 12 times (every 1200 seconds in the 4 hour long trace). Without the NSR module, the plain data center simulation (without rerouting scheme) took close to 55 seconds. Thus each NSR improvement roughly took 11 seconds ($(180 - 55)/12 \approx 11$).

In 16.66 hour long simulation run with "synthetic data traffic" trace (VI-B), consisting of 3200 VMs and 1024 servers, the time taken per simulation run was close to 2000 seconds. The NSR improvement was invoked 50 times (every 1200 seconds in the 16.66 hour long trace). Without the NSR module, the plain data center simulation (without rerouting scheme) took close to 620 seconds to complete. Because of the increased number of VMs and bigger size of the data center, each NSR improvement run took roughly 28 seconds ($(2000 - 620)/50 \approx 28$). All the simulations were run on a Intel Quad Core Machine with each core clocked at 2.66GHz.

Finally, we note that our method does not need to perform real-time monitoring of flow sizes and link utilizations (both of which can be an overhead). Since we keep track of reservations made for each VM of each tenant on all links, we know the residual (unreserved) link capacities analytically.

VIII. Related Work

The increase in the number of multi-tenant data centers have resulted in a significant interest in performance isolation of tenants through network virtualization. While [3], [2], [4], [5] provide absolute bandwidth guarantees for VMs of a tenant, [6], [7] provide weighted-fair link bandwidth sharing among different tenants. As mentioned in [16], while bandwidth guarantees for the VMs are preferred by clients because it provides predictable network performance for the tenants' applications, it may result in low average link utilization when tenants' bandwidth requirements are lower than the reserved levels. On the other hand, data center operators prefer proportional sharing of link bandwidth which allows higher average link utilization, and when the data center network is oversubscribed, it provides link-level fairness among tenants.

In this work, by using Network Satisfaction Ratio (NSR) and time-varying reservation request, we propose a network reservation solution that provides the benefits of both the above approaches. By continuously striving to achieve an NSR of 1 for each tenant, our method tries to provide bandwidth reservation as close to the desired level of the tenant. On the

other hand, by allowing time-varying network reservation, we provide a mechanism for tenants to match their bandwidth request close to their current usage. Finally, by maximizing the minimum NSR, we provide fairness across tenants, which is useful when the network is oversubscribed. The key difference in our solution as compared to the above reservation schemes is the notion of NSR, which allows us to design a method that is aware of tenants' desired rates, and also allows us to compare network reservations across tenants.

Among the above network reservation methods, our work is closest to [3] and [2]. We use the virtual cluster as the network reservation abstraction, and (a slightly modified) initial placement scheme from [3]. However, our work differs from [3] in the following three points: (a) We allow tenants to modify their placement over time, and continually modify the network reservation (flow routing and VM placement) according to the tenants' requests, whereas, [3] only considers initial placement; (b) (as mentioned above) we fairly distribute the network resources among tenants in case of a network oversubscription, (c) our algorithms work over topologies where there are multiple paths among VMs, whereas the algorithm in [3] is designed for tree topology. Although [2] allows time varying network reservation and routing over non-tree topologies, it does not provide fairness across tenants, and it considers clique abstraction for network reservation for a tenant. The clique abstraction allows a tenant to specify bandwidth requirement between all pairs of VMs. Although it provides more flexibility to the tenants to specify network requirement, a clique is typically more difficult to provision in a network due to its larger number of virtual links.

The overall goal of our work is also close to [17], which proposes bandwidth allocation methods to fairly share network bandwidth among tenant. In [17], the authors point out three desirable characteristics of a bandwidth allocation scheme: minimum absolute bandwidth guarantee for every VM, high network utilization, and sharing the network among tenants in proportion to their payments. Further, the paper points out fundamental tradeoff between these requirements, and presents methods that satisfy some of the requirements. Although our algorithm aims to share the network in proportion to tenants' payments (assuming their payments are proportional to their desired network bandwidths), our algorithm differs from [17] in two crucial ways. First, we directly take as an input the desired bandwidth for each VM, and try to fairly reserve as high a fraction of that bandwidth as possible, whereas, [17] provides VM bandwidth as a function of the VM's weight and total number of VMs of the tenant. We believe our reservation interface provides a simpler abstraction for the tenants. Second, our algorithm considers flow routing and VM placement as part of the reservation scheme, but [17] relies on underlying network routing and does not modify VM placement. Modifying flow routing and VM placement can be useful when the requirements of the tenants vary over time.

The work presented in the current paper complements our earlier work on optimal network-aware initial placement of VMs to save network power [15], steady state network-aware migratons of VMs for network congestion resolution [13] and a QoS scheme for VM migrations [18].

IX. Concluding Remarks

In this paper, we have presented a novel method for fairly managing tenants' network reservations in oversubscribed cloud data centers. In trace-driven simulations, our method shows significant benefit in terms of fairness across tenants and number of accepted reservation requests, as compared to baseline schemes. We now mention some directions for future work. First, the NSR metric that we use in this work can be extended to include other network performance indicators and to capture a tenant's level of satisfaction in a better way. Second, in this work we primarily considered reservation for network resources, and it would be useful to study the joint problem of fairly managing network, compute, storage and memory resource allocations. Finally, in addition to virtual cluster and clique abstractions, it would be interesting to investigate other networks reservation abstractions that are tailored for common types of data center applications.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*, 2004.

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

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

[4] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *USENIX Workshop on I/O Virtualization*, 2011.

[5] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: A cloud networking platform for enterprise applications," in *ACM SoCC*, 2011.

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

[7] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese, "Netshare and stochastic netshare: predictable bandwidth allocation for data centers," *ACM Computer Communication Review*, vol. 42, no. 3, 2012.

[8] D. Xie, N. Ding, Y. C. Hu, and R. R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," in *ACM SIGCOMM*, 2012.

[9] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, "A flexible model for resource management in virtual private networks," in *ACM SIGCOMM*, 1999.

[10] Y. Chen, S. Alspaugh, and R. Katz, "Interactive Analytical Processing in Big Data Systems: A CrossIndustry Study of MapReduce Workloads," in *VLDB*, 2012.

[11] "Hadoop 1.1.1 Documentation." [Online]. Available: http://hadoop.apache.org/docs/r1.1.1/

[12] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, " The Case for Evaluating MapReduce Performance Using Workload Suites," in *MASCOTS*, 2011.

[13] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware Steady State VM Management for Data Centers," in *IFIP Networking*, 2012.

[14] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "Oflops: an open framework for openflow switch evaluation," in *Proceedings of the 13th international conference on Passive and Active Measurement (PAM)*, 2012.

[15] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "Vmflow: Leveraging vm mobility to reduce network power costs in data centers," in *IFIP Networking*, 2011.

[16] F. Bari, R. Boutaba, Esteves, M. Podlesny, G. Rabbani, Q. Zhang, F. Zhani, and L.Granville, "Data center network virtualization: A survey," *IEEE Communications Surveys and Tutorials*.

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

[18] V. Mann, A. Vishnoi, A. Iyer, and P. Bhattacharya, "VMPatrol: Dynamic and Automated QoS for Virtual Machine Migrations," in *CNSM*, 2012.