Using Empirical Estimates of Effective Bandwidth in Network-Aware Placement of Virtual Machines in Datacenters

Runxin Wang, *Student Member, IEEE*, Juliano Araujo Wickboldt, *Student Member, IEEE*, Rafael Pereira Esteves, *Member, IEEE*, Lei Shi, *Member, IEEE*, Brendan Jennings, *Member, IEEE*, and Lisandro Zambenedetti Granville, *Member, IEEE*

***Abstract*—Datacenter operators are increasingly deploying virtualization platforms to improve resource usage efficiency and to simplify the management of tenant applications. Although there are significant efficiency gains to be made, predicting performance becomes a major challenge, especially given the difficulty of allocating datacenter network bandwidth to multitier applications, which generate highly variable traffic flows between their constituent software components. Static bandwidth allocation based on peak traffic rates ensures SLA compliance at the cost of significant overprovisioning, while allocation based on mean traffic rates ensures efficient usage of bandwidth at the cost of QoS violations. We describe MAPLE, a network-aware VM ensemble placement system that uses empirical estimations of the effective bandwidth required between servers to ensure that QoS violations are within targets specified in the SLAs for tenant applications. Moreover, we describe an extended version of MAPLE, termed MAPLEx, which allows the specification of anticolocation constraints relating to the placement of application VMs. Experimental results, obtained using an emulated datacenter, show that, in contrast to the Oktopus network-aware VM placement system, MAPLE can allocate computing and network resources in a manner that balances efficiency of resource utilization with performance predictability.**

***Index Terms*—Virtualization, datacenter management, resource management, effective bandwidth, virtual machine placement.**

I. Introduction

MANY modern datacenters are composed of a massive number of networked commodity servers that are virtualized to provide computing resources on demand to a large number of tenant applications. Without an adequate network infrastructure, datacenters cannot properly support the performance requirements of many mission critical multi-tier applications. Studies [1], [2] have identified that unpredictable network performance in datacenters can become a bottleneck to many cloud-hosted applications, with many researchers having proposed solutions [3]–[11].

A prevalent approach in the industry to achieve relatively predictable datacenter networks is to strictly reserve bandwidths for tenants' Virtual Machine (VM) ensembles. We use the term 'ensemble' to refer to the group of VM instances involved in the delivery of an application's functionality. This is the case, for example, in Amazon EC2 [12] and Rackspace [13] Infrastructure-as-a-Service (IaaS) offerings. However, strict bandwidth reservation does not efficiently utilize network resources—unused bandwidth is wasted during periods when VM traffic demands are below the provisioned peak rates. Nevertheless, overprovisioning is a pragmatic approach, given that it is extremely difficult for tenants to predict accurately and in advance inter-VM bandwidth requirements for their VM ensembles. Thus, tenants can be provided with weak statistical guarantees that Quality-of-Service (QoS) targets in their Service Level Agreements (SLAs) will be met. However, VM placement systems relying on overprovisioning typically lack the capability to adjust the VM bandwidth allocations following initial placement.

In response to overprovisioning issues in datacenters, recent work on datacenter network management has focused on allocating network resources to tenants' VM ensembles at optimal cost by implementing 'network-aware' VM placement algorithms and accompanying traffic control mechanisms [7]–[11]. Often, these approaches seek to minimize bandwidth wastage by reallocating bandwidth not being used by a VM to other VMs in the same cluster. However, these approaches largely focus on statistically guaranteeing the throughput performance of VMs; they do not address the potential for increased delays due to transient overloads of network links that may occur as a result of a less strict bandwidth allocation regime. In this paper, we address the problem of efficiently utilizing datacenter network resources while ensuring that QoS delay targets specified in SLAs are met. In particular, we seek to support probabilistic QoS targets expressed in the following manner: "no more than 2% of packets should be delayed by more than 50 ms."

Manuscript received June 3, 2015, revised October 26, 2015, accepted January 9, 2016. Date of publication February 15, 2016; date of current version June 8, 2016. This work was supported in part by the Science Foundation Ireland via the CONNECT research centre (Grant 13/RC/2077) and the Research Brazil Ireland project (Grant 13/ISCA/2843); in part by the Irish Higher Education Authority under the Programme for Research in Third-Level Institutions (PRTLI) Cycle 5 (co-funded under the European Regional Development Fund) via the Telecommunications Graduate Initiative project; in part by the Irish Research Council via Grant ELEVATEPD/2013/26; and in part by the European Commission via the SOLAS IAPP FP7 project (Grant 612480). The associate editor coordinating the review of this paper and approving it for publication was P. Owezarski.

R. Wang, L. Shi, and B. Jennings are with the Telecommunications Software and Systems Group (TSSG), Waterford Institute of Technology (WIT), Waterford, Ireland (e-mail: rwang@tssg.org; lshi@tssg.org; bjennings@wit.ie).

J. A. Wickboldt, R. P. Esteves, and L. Z. Granville are with the Federal University of Rio Grande do Sul (UFRGS), Rio Grande do Sul, Brazil.

Digital Object Identifier 10.1109/TNSM.2016.2530309

Our approach is to provide a network-aware VM placement scheme in which VMs within an ensemble that need to be placed on different servers are placed in a manner that ensures that the "effective bandwidth" available on the network path between servers is sufficient. Kelly [14] defined effective bandwidth as the "minimum amount of bandwidth required by a traffic source to maintain specified QoS targets." We require neither *a priori* bandwidth reservation nor the implementation of traffic control mechanisms in server hypervisors. Once a VM ensemble is placed, VMs are enabled to asynchronously reach their peak throughputs. We rely on the use, in the admission decision, of empirically computed estimates of effective bandwidth on the network paths to ensure that the likelihood of SLA violation is minimal. This approach allows the datacenter provider to specify QoS targets for hosted applications in terms of delay, whilst ensuring that network resources are utilized efficiently.

This article is an extended version of our earlier work [15], which introduced the MAPLE system. MAPLE has been developed to accomplish the following objectives: 1) provision of predictable network performance to tenant's VM ensembles; 2) optimal joint-allocation of network and computing resources; and 3) satisfaction of applications' QoS targets. The design addresses the challenges in applying effective bandwidth techniques to manage datacenter networks. First, we design a network-aware VM placement algorithm that uses the effective bandwidth technique to produce optimal VM placement solutions in a timely manner. Second, we seek to decentralize the effective bandwidth estimations to improve the run-time performance.

The main differences between the material provided here and Wang et al. [15] are as follows:

1) we describe an extended version of MAPLE, which we term MAPLEx, that facilitates specification of anti-colocation constraints [16], [17] for the placement of ensemble VMs. Our formulation of anti-colocation constraints allows specification of two forms of anti-colocation: 1) VMs from the same ensemble not being allowed to be placed on the same server; and 2) VMs from an ensemble not being allowed to be placed on a server on which VMs from other ensembles are already hosted. The former are often used in IaaS platforms and enterprise datacenters to ensure fault-tolerance and high availability for ensembles realizing, for example, high-volume web services [18]. The latter ensure that VMs in a given ensemble are not placed on a PM together with VMs from other ensembles. Such constraints would be used to provide an protection against security threats that, in an IaaS environment, may emanate from co-located VMs that might exploit hypervisor vulnerabilities [19], [20]. A comparison of MAPLE and MAPLEx confirms that the introduction of anti-colocation constraints can lead to sub-optimal rates of successful VM placement in resource-constrained scenarios;
2) we perform experiments on a larger test-bed, comprising 4 physical servers (Wang et al. [15] used a single server), which allows emulation of a datacenter having 280 VM slots, communicating with each other via a network comprised of two layers of virtual switches and one root physical switch. More significantly, whilst [15] relied on a synthetic traffic generation application to generate workloads, for the experiments described here we use a combination of three applications—the original synthetic application based on SCP, along with a Hadoop that runs word count on distributed documents, and a combination of YCSB (as a data client) and Cassandra (as a data server). Finally, our experimental scenarios also model the withdrawal of VM ensembles from the systems after a specified time. The enhanced test-bed thus provides a more realistic approximation of a production datacenter environment.

The paper is structured as follows. In §II we review related work on network bandwidth allocation systems in datacenters. The concept of effective bandwidth is introduced in §III. In §IV and §V we respectively describe MAPLE system design, and the network-aware VM placement algorithm and its algorithmic variant, MAPLEx, that copes with anti-colocation constraints. Experimental evaluation is presented in §VI, where MAPLE/MAPLEx are evaluated and compared to Oktopus [6]. Finally §VII provides concluding remarks and discusses topics for future work.

II. Related Work

A number of recent publications have focused on network resource allocation in datacenters. However, effective bandwidth estimation has not yet been applied for this purpose. Here we focus on recent proposals that are relevant to our approach. For a more complete view of the literature, we refer to Bari et al. [21] as well as Jennings and Stadler [22].

Guo et al. [23] describe SecondNet, which allows tenants to select between QoS classes for their applications. Their proposal focuses on prioritizing traffic, but it does not deal with specific QoS delay targets of the kind addressed by MAPLE. In SecondNet, virtual datacenters (VDCs) are leased to tenants, who are able to specify bandwidth guarantees for their VDCs by using traffic matrices. The system then applies a port-switching source routing mechanism to realize bandwidth guarantees. This moves the bandwidth reservation tasks from switches to hypervisors. Kanagavelu et al. [24] also address routing of inter-VM traffic. They propose a scheme in which VMs are placed on servers having multiple link disjoint paths to other servers hosting VMs, arguing that this can both help reduce the probability of congested links whilst improving resilience in cases where links fail.

Ballani et al. [6] proposed Oktopus, in which VM ensembles are placed based on virtual cluster abstractions. In their work, all VMs are modeled as being connected to a single virtual switch. A datacenter tenant can choose the abstraction and the degree of the over-subscription of the virtual cluster based on the communication patterns of the application VMs the tenant plans to deploy. Instead of reserving the required bandwidth, Oktopus applies a greedy algorithm to map a virtual cluster onto a physical datacenter network in order to save bandwidth that VM pairs cannot use. The bandwidth required by each VM to connect to the virtual switch can be either the

expected mean bandwidth of the traffic generated by the VM or the expected peak bandwidth. Angel et al. [25] also address the creation of virtual clusters, but extend the concept to VDCs realized through both a virtual network and one or more virtual network appliances. They argue that much of the application performance degradation in such scenarios can arise from congested appliances, which often exhibit varying service times for requests based on the request content. They describe Pulsar, a logically centralized controller that estimates applications' vary demands and the available capacity of appliances and allocates resources based on these estimates according to flexible resource allocation policies. Whilst controlling the allocation of appliance resources is clearly an important issue the work does not address satisfaction of explicit QoS targets.

Lam et al. [7] describe NetShare, which assigns bandwidth by virtualizing a datacenter network using a statistical multiplexing mechanism. Network links are shared in the level of services, applications, or corporate groups, rather than via single VM pairs. In this way, one service, application, or group cannot consume more available bandwidth by opening more connections. However, this weight-based bandwidth allocation approach does not provide bandwidth guarantees to VM pairs.

Popa et al. [9] developed ElasticSwitch, which can effectively provide bandwidth guarantees to VMs in a work-conserving way, since it does not require strict bandwidth reservation. ElasticSwitch comprises two functional layers to realize its design objectives: the guarantee partitioning layer ensures that the bandwidth guarantees of each VM pair connection are met, and the rate allocation layer observes the actual bandwidth each connection needs and reallocates some unused bandwidth to the active connection in order to improve link utilization.

Wuhib et al. [11] address the placement of VDCs in a datacenter environment in order to optimally allocate resources to meet four management objectives—balanced load, energy efficiency, fair allocation, and service differentiation. Notably, they present a distributed solution based on the use of a gossip protocol, which they show to be both effective and highly scalable, handling up to 100,000 nodes for the considered management objectives. However, they provide only for static allocation of (peak) inter-VM bandwidth, which may lead to over-provisioning or QoS violations.

LaCurts et al. [10] describe Choreo, a datacenter network management system that is based on application profiling. Choreo has three sub-systems: a measurement component to obtain inter-VM traffic rates, a component to profile the data transfer characteristics of a distributed application, and a VM placement algorithm. The application traffic profiling subsystem mainly involves profiling the application to find its network demands and measuring the network to obtain the available bandwidths between VM pairs. Based on this information, VMs can be optimally placed to achieve predictable network performance.

Breitgand and Epstein [16] directly target the problem of placing VMs on physical machines in a manner that minimizes the potential for groups of VMs contending for bandwidth on egress links to lead to congestion and, thus, potentially to violation of service SLAs. In an approach similar to that described here, they take into account the statistical multiplexing effect to avoid under-allocating VMs to physical machines. To do so they cast VM placement as a stochastic bin packing problem in which VMs' bandwidth requirements are modelled as random variables. This approach requires prior knowledge of VM traffic generation patterns, which contrasts with MAPLE's measurement-based approach, which does not require prior knowledge and can adapt to changing traffic patterns over the course of VMs' lifetimes.

Biran et al. [26] also address network-aware VM placement; their Min Cut Ratio-aware VM Placement (MCRVMP) formulation and heuristics incorporates constraints evolving from complex network topologies and dynamic routing schemes, but also take into account the time varying nature of traffic demands in a manner that finds placements that have sufficient spare capacity across the network to absorb unpredicted traffic bursts. Other works, including Meng et al. [5], Wang et al. [27], and Duong-Ba et al. [28] also address the modeling of VM traffic in order to attain optimal VM placement or consolidations that save bandwidth consumption. However, these works do not directly address how to ensure that delay-based QoS targets of application SLAs can be satisfied.

III. Effective Bandwidth Review and Residual Bandwidth Estimation

A. *Effective Bandwidth and Its Estimation*

Effective bandwidth is the minimum amount of bandwidth required by a traffic source (for example, a VM or an application) to maintain specified QoS targets. In a communications network, the effective bandwidth of traffic sources depends not only on the sources themselves, but also on the whole set of systems, including link capacity, traffic characteristics, and the QoS target [29]. For example, consider the scenario of a link with capacity of 1 Gbps and two VMs that generate traffic with mean throughput of 300 Mbps and peak throughput of 550 Mbps, where the probability of peak throughput is 10% for each VM, and the QoS target specifies that no more than 5% of the traffic suffers delays higher than 50 ms. The question that arises is whether the given link can accommodate these two VMs. If the two VMs reach the peak throughput at the same time, the aggregated throughput would be 1100 Mbps, exceeding the total link capacity of 1 Gbps. Assuming there is a shaping policy, the exceeding traffic would be delayed more than 50 ms. However, the probability of this situation actually happening is only 1% (assuming VMs are independent). Therefore, the QoS target is not violated, and the link can accommodate the VMs. To allow each VM to asynchronously reach its peak, the allocated bandwidth of each VM should be higher than its mean throughput. Moreover, it is not necessary to allocate peak throughput to the VMs because the chance that they both reach peak is small enough to statistically guarantee that the QoS target is not violated. This example indicates that the effective bandwidth lies somewhere between the source's mean throughput and peak throughput [14].

Effective bandwidth can be analytically estimated through the application of large deviation theory [29], or via more

computationally simple approximations based on fluid-flow and stationary models [30]. However, in MAPLE, we employ an empirical approach based on the analysis of traffic traces collected at each server. At set intervals, we estimate the effective bandwidth for a given delay-based QoS target for the aggregated traffic generated by the server in question over the trace duration. This value, plus the peak rate of the VM that is a candidate to be placed on the server, are compared to the available bandwidth so to assess whether the VM can be placed on that server. The effective bandwidth estimation technique is taken from Davy et al. [31] and is summarized as follows.

Let $delay_{max}$ be the nominal maximum delay and let p_{delay} be the percentage of traffic that can exhibit delay greater than $delay_{max}$. We define the effective bandwidth R_{eff} of a traffic source for delay QoS target $(delay_{max}, p_{delay})$ as the minimal rate R such that if we simulate a FIFO queue with unlimited buffer and processing rate R, the percentage of traffic that will exhibit delay greater than $delay_{max}$ will be less than p_{delay}. To estimate the effective bandwidth of a particular traffic source on the network, we take a recorded packet trace of that source. We observe that if we simulate a FIFO queue (initially assumed to be empty) with the same input traffic trace $\{T_M\}$ for different process rates $R_1 > R_2$ and estimate the percentages p_1 and p_2 of traffic delayed more than $delay_{max}$ for different rates respectively, then $p_1 \leq p_2$. This means that the percentage of traffic, p, delayed more than $delay_{max}$ is a monotonically decreasing function of processing rate R. Based on this observation, we employ a simple bisection algorithm for a recorded packet trace to find the minimal value of a queue rate such that the percentage of traffic delayed more than $delay_{max}$ is less than p_{delay} (a full specification of this algorithm and analysis of its estimation accuracy is provided by Davy et al. [31], [32]).

B. Residual Bandwidth Adjustment

Existing network-aware VM placement techniques often apply the hose model [2] to control the maximum data rates that the datacenter-hosted VMs can reach. With the hose model, bandwidth is allocated on a per-VM basis. The residual bandwidth of a server is then calculated as the server's link capacity minus the sum of the bandwidths allocated to VMs placed on the server. If a server's outgoing link capacity is C, with N VMs placed on it, each allocated a bandwidth of B, then the residual bandwidth of this server is taken to be $(C - N \cdot B)$. As described above, if VM bandwidth allocations are based on peak expected traffic rates, this approach is likely to lead to significant underutilization of the available bandwidth and the potential rejection of VM admissions that could be successfully provisioned.

In MAPLE, on the other hand, we apply effective bandwidth estimation to determine the minimum bandwidth required to provision the allocated VMs whilst meeting QoS targets. Accordingly, the residual bandwidth of a server is calculated as $(C - R_{eff})$, where R_{eff} in this case refers to the corresponding empirically estimated effective bandwidth of the aggregated traffic currently transferred across the link. Fig. 1 illustrates the two different methods to calculate the residual bandwidth of a server that is hosting two VMs and has link capacity of

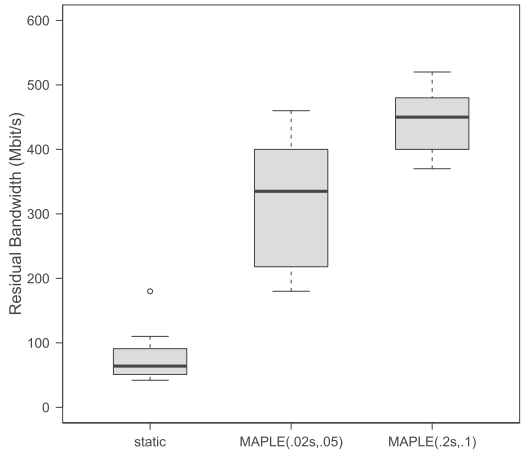


Fig. 1. Box plots of the residual bandwidth of a server having egress link capacity of 1 Gbps, obtained using our test-bed. The value inside the brackets indicate the QoS target, the first term being the target delay, the second the maximum percentage of violations of this delay target. When peak bandwidths are reserved for each VM, little residual bandwidth remains. In contrast, when bandwidth is allocated based on effective bandwidth, the available residual bandwidth is significantly higher, with the value depending on the stringency of the QoS target.

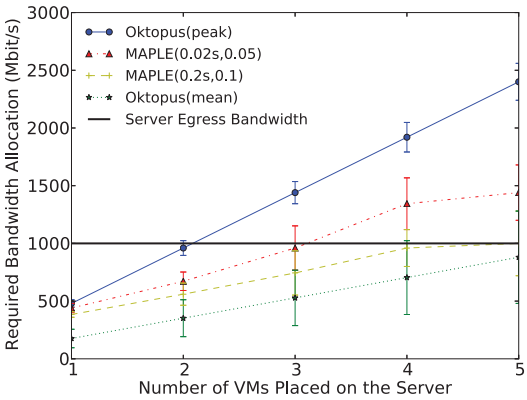


Fig. 2. As additional VMs are placed on a server, the required egress bandwidth increases. If peak or mean throughputs per VM are statically allocated as in Oktopus, this bandwidth increases linearly. If effective bandwidth estimations are used, the rate of increase reduces because of the smoothing effect of the statistical multiplexing of traffic from numerous sources. MAPLE can accommodate more VMs on each server (whilst satisfying QoS targets) than Oktopus (whilst allocating based on peak throughput).

1000 Mbps. Static reservation refers to the method that reserves maximum throughput (on average 940 Mbps) for the VMs and computes the residual bandwidth accordingly, whereas effective adjustment refers to adjusting the residual bandwidth based on the effective bandwidth estimates. In this case, the latter determines that on average an aggregated bandwidth of 640 Mbps is sufficient for the two VMs for achieving a given QoS target (0.02 s, 0.05) – no more than 5% of packets suffer delay longer than 0.02 s, whereas given a lower QoS target (0.1 s, 0.2) the residual bandwidth is higher.

Fig. 2 illustrates how an effective bandwidth technique can lead to more efficient utilization of computing resources in comparison to allocation of peak expected bandwidth when hosted VMs heavily use a server's egress bandwidth. We emulate the scenario where 5 VMs share an egress bandwidth of

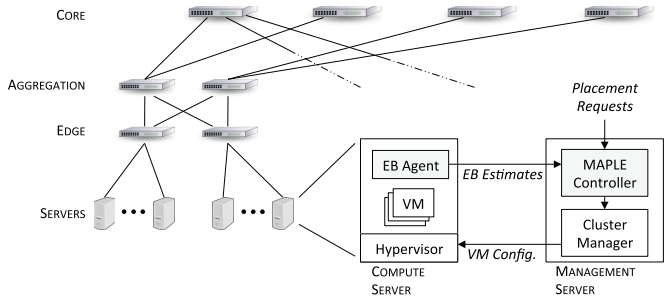


Fig. 3. The MAPLE system architecture. MAPLE is comprised of Effective Bandwidth Agents (EB Agents) residing in each server and a MAPLE Controller residing in a management server. EB Agents send effective bandwidth estimates upon request to the MAPLE Controller. The MAPLE Controller processes VM placement requests from tenants and instructs the Cluster Manager, which in turn configures VMs on the selected servers. Whilst the figures illustrates a Fat Tree datacenter topology (see Portland [34]), MAPLE is agnostic of the topology since it collects bandwidth estimates at servers only.

1 Gbps. Given a QoS target of $(0.02\ s, 0.05)$, with more VMs arriving (recall that all VMs generate SCP traffic), it will be, as illustrated in Fig. 2, impossible to place more than 2 VMs on the server if the peak bandwidth of each VM must be strictly provisioned for their use. However, using MAPLE, it is possible to place up to 3 VMs on the server without violating the QoS target $(0.02\ s, 0.05)$. Given a lower QoS target of $(0.2\ s, 0.1)$, it is possible to host up to 5 VMs on the server, as shown in Fig. 2. Placing VMs on the basis of strictly provisioning for VMs' mean throughput would allow the placement of more VMs on the server. However, as discussed below, this would be at the cost of a significant number of QoS violations.

IV. MAPLE Architecture

MAPLE and its extension, MAPLEx, are designed to manage the joint allocation of network and computing resources in datacenter clusters to provision VM ensemble requests from tenants in a manner that provides statistical guarantees that QoS targets specified in SLAs are satisfied. Based on the design objectives described in §I, MAPLE has two main components. As illustrated in Fig. 3, these components are: 1) the MAPLE Controller, which is deployed on a cluster management server and interacts with a cluster manager such as VMware vCenter [33]; and 2) effective bandwidth estimation agents (EB Agents) deployed on each server, which analyze outgoing aggregated traffic to estimate its effective bandwidth and periodically send this information to the MAPLE Controller. We now describe these components in more detail.

A. EB Agents

EB Agents are deployed on every server in which MAPLE can place VMs. They are responsible for collecting traces of traffic emanating from the server using utilities such as tshark (a command line version of the Wireshark network analyzer [35]). Traces are then used to estimate effective bandwidth using the approach described in §III. EB Agents collect traces at K minute intervals, each time storing S minute long traces.

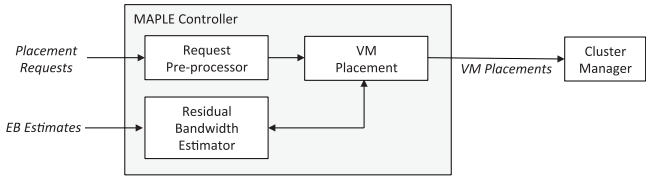
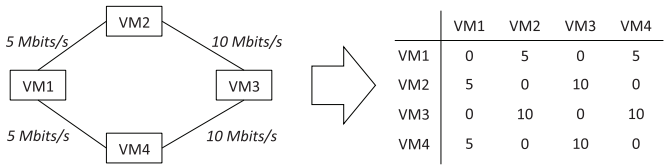


Fig. 4. The MAPLE Controller. Incoming requests for VM ensemble placement are first preprocessed to account for VMs that need to be placed together. VM placement decisions are then made taking into account the estimated available residual bandwidth at each server. Identified placements are passed to a Cluster Manager for configuration.



	VM1	VM2	VM3	VM4
VM1	0	5	0	5
VM2	5	0	10	0
VM3	0	10	0	10
VM4	5	0	10	0

Fig. 5. A tenant VM ensemble topology specification indicates the expected peak traffic rates between VMs comprising the tenant application, together with the corresponding traffic matrix. In cases where VMs are to be co-located on the same server, the Request Preprocessor in the MAPLE Controller groups these VMs and generates a simplified traffic matrix.

The effective bandwidth for the QoS target(s) specified by the MAPLE Controller is estimated for each trace. Whenever the MAPLE Controller requests an effective bandwidth estimate, the EB agent selects the maximum estimate from the last five stored estimates.

The effective bandwidth estimation process executed by EB Agents is an offline process, in the sense that it is not executed every time a VM ensemble placement request arrives. Given that, we have not undertaken a detailed study of its overhead, but our experience with our experimental evaluation of MAPLE indicates that EB Agents utilize an insignificant portion of the resources of the PM on which they are hosted. Moreover, this overhead can be controlled by adjusting the values of K and S. Finally, since all trace analysis is done locally on the server, there is minimal overhead incurred in EB Agent to MAPLE Controller communications.

B. MAPLE Controller

The MAPLE Controller handles incoming requests for VM ensemble placements, deciding on a one-by-one basis if a given request can be accepted and computing the placement if it can. As depicted in Fig. 4, it is comprised of three functional entities, which we now describe.

1) Request Preprocessor: The Request Preprocessor receives VM ensemble placement requests that specify one of a small number of QoS classes offered by the datacenter provider. These QoS targets are specified in terms of packet delays rather than simply in terms of overall throughput levels. As depicted in Fig. 5, VM ensemble placement requests prescribe a topology of VMs that comprise an application and indicate peak traffic flow rates between VMs on a pairwise basis. For simplicity, we assume in this paper that the same amount of bandwidth is utilized in both directions between a VM pair; thus the total peak traffic rate for a VM in a

server is simply the sum of the rates in the relevant row of the traffic matrix. The Request Preprocessor also performs some preprocessing aimed to optimize the subsequent VM placement. It allows placement requests to specify that particular VMs should be placed together on the same server, given the expectation that they will interact heavily with one another. Given this, the Preprocessor and MAPLE Controller treat such VMs as a single VM, thus simplifying the traffic matrix accordingly. The opposite scenario, in which specified VMs are not to be co-located, is handled by MAPLEx, as described in §V-A.

2) Residual Bandwidth Estimator: This entity manages and queries EB Agents deployed on the compute servers under the control of the MAPLE system. Whenever a VM ensemble placement request arrives, the Residual Bandwidth Estimator queries a number of servers, who reply with their effective bandwidth estimate. A number of options are possible to govern which servers are queried. For example, a number of servers can be sampled at random, with the number being calculated as a function of the overall occupancy of the datacenter clusters—in lightly loaded clusters fewer servers would need to be queried in order to find a viable placement. Alternatively, only servers already hosting VMs could be sampled initially so that VMs are consolidated for energy optimization purposes. For simplicity, we assume here that all servers in the cluster are queried; alternative strategies will be explored in future work. When effective bandwidth estimates are received, the residual bandwidth as seen by the servers is calculated and the server IDs and associated residual bandwidth estimates are passed to the VM Placement entity.

3) VM Placement: This entity takes as input the preprocessed VM ensemble placement requests. It requests the Residual Bandwidth Estimator to provide it with a set of candidate servers for the placement and the estimates of residual bandwidth available on the egress links of those servers. It then executes the MAPLE or MAPLEx network-aware VM placement algorithms specified in §V. If the VM ensemble can be placed, the placement details are passed to the Cluster Manager that applies configurations on the servers accordingly.

V. Network-Aware VM Placement Algorithms

The MAPLE Controller seeks to minimize both the nominally allocated network bandwidth and the number of servers in which VMs are placed, such that QoS targets are met. As multi-objective VM placement problems are known to be NP-hard [5], [16], [27], in this work MAPLE uses the heuristic algorithm specified in Alg. 1 and Alg. 2 to produce VM ensemble placement results in a timely manner.

The algorithm employs the First Fit Decreasing (FFD) approach to search for VM placement solutions—FFD has been widely applied to VM placement problems [16], [27], [36]. The algorithm first sorts servers in increasing order of their β values using Eqn. 1 and commences searching. The approach is known as "first fit" because it stops searching once it finds the first feasible placement. FFD approaches generally give sub-optimal results, but are able to find feasible solutions in a timely manner. In our algorithm, VM placements are optimized in the sense that, since servers are sorted in increasing order of residual bandwidth, the first fit placements are most likely found on servers that have relatively smaller residual bandwidths. In this way, VMs are placed into servers until there is no available computing capacity or sufficient residual bandwidth to meet

Algorithm 1. MAPLE VM Ensemble Placement

Input: N, B, $node$
Output: True or False
1: $done$=False, $subtrees$=$node$.subtrees
2: **if** sizeof($subtrees$) = 0 **then**
3: **if** $node$.remainSlots $\geq$ sizeof(N) **and** $node$.remainBW $\geq$ sizeof(N)*B **then**
4: allocate N to $node$
5: return True
6: **else**
7: return False
8: **end if**
9: **else**
10: **sort** $subtrees$ based on the β values Eqn. 1
11: **for** $subtree$ in $subtrees$ **do**
12: $done$ = MAPLE(N,B,$subtree$)
13: **if** $done$ = True **then**
14: return $done$
15: **end if**
16: **end for**
17: **end if**
18: **if** $done$ == False **then**
19: $tail$ = allocBetweenNodes(N,B,$subtrees$)
20: **end if**
21: **if** $tail$!= N **then**
22: $done$ = MAPLE($tail$,B,$node$)
23: **end if**
24: return $done$

Algorithm 2. allocBetweenNodes function

Input: N, B, $nodes$
Output: $tail$
1: $tail$=N
2: **sort** $nodes$ based on the β values eq. 1
3: **for** $node$ in $nodes$ **do**
4: **if** $node$.remainSlot = 0 **then**
5: continue
6: **end if**
7: m=$node$.remainSlot, n=sizeof(N), t=min(m,$n-m$)
8: **if** $node$.remainBW $\geq$ t*B **then**
9: $head$=N[0 : m], $tail$=N[m : n]
10: **if** MAPLE($head$,B,$node$) = True **then**
11: return $tail$
12: **else**
13: continue
14: **end if**
15: **end if**
16: **end for**
17: return $tail$

the QoS target. This serves to both minimize the number of servers used and to ensure that the overall bandwidth nominally allocated to VMs is minimized.

The β metric, computed using Eqn. 1, is used to ensure that the MAPLE algorithm sorts servers in increasing order of residual bandwidth. The coefficient $g \in [0, 1)$ is a configurable control parameter that influences the degree to which β reflects the residual number of VM slots. For example, when $g = 0$, the residual number of VM slots in the servers will have no effect; otherwise, when servers have the same level of residual bandwidths, those with fewer remaining VM slots will be searched first. Note that in our experimental evaluation we have used $g = 0.5$. In previous work [36] we investigated the approach of formulating a metric similar to β as the square root of the sum of squares of the components of interest, verifying this approach's effectiveness in terms of resource utilization efficiency; other authors, for example Sheikhalishahi et al. [37] have adopted a similar formulation.

$$\beta = \sqrt{\left(\frac{\text{residual\_BW}}{\text{total\_BW}}\right)^2 + g \cdot \left(\frac{|\text{residual\_slots}|}{|\text{all\_slots}|}\right)^2} \qquad (1)$$

Notice that both the residual bandwidth level and residual slot level are expressed as proportional values. This reflects the fact that the absolute values of the residual bandwidth and number of slots can vary significantly, leading one to dominate the other.

The input to the MAPLE algorithm is a VM ensemble request $< N, B >$, where N is a set of VMs in request and B is the set of associated expected peak bandwidth utilization. To simplify the presentation of the algorithm without losing generality, we assume that all VMs expect the same peak bandwidth utilization. The algorithm assumes that the datacenter topology can be represented by a tree structure, which is the case for fat-tree topologies typically used in datacenter networks [1], [34], [38]. When n requests arrive at a time t, these requests will be sorted based on their β values using Eqn. 1, and then processed one by one. The algorithm handles three cases:

1) Case I (shown in lines 2–8)—when a given node is a server (the lowest subtree that has no further subtree, as shown in line 2), MAPLE attempts to allocate the entire VM ensemble placement request into the same server;
2) Case II (addressed in lines 10–16)—if a node has subtrees, the subtrees will be sorted increasingly based on their β values, and MAPLE attempts to allocate the entire VM ensemble placement request into the same subtree. Notice that here we assume that the routing cost is relatively less expensive when VMs are located in the same subtree;
3) Case III (addressed in lines 18–23)—when the algorithm cannot find any subtree that can host the entire VM ensemble, it attempts to allocate the requested VMs into different subtrees. VMs are divided into two groups: *head* and *tail*. Function allocBetweenNodes() (specified in Alg. 2) returns *tail*, which is the group of VMs not yet allocated after it successfully allocated the VMs in the *head* group; otherwise, Alg. 2 returns the original input, indicating that the input VMs cannot be allocated into different subtrees.

In an approach similar to that taken by Oktopus [6], whenever a VM ensemble request cannot be entirely placed into one

Algorithm 3. MAPLEx VM Ensemble Placement

Input: N, B, x, $node$
Output: True or False
1: $done$=False, $subtrees$=$node$.subtrees
2: **if** sizeof($subtrees$) = 0 **then**
3: n = sizeof(N)
4: **if** $node$.remainSlots $\geq x$ **and** $node$.remainBW $\geq$ n*B **then**
5: $X := \{VM_i \in N | 0 \leq i < min(n, x)\}$
6: allocate X to $node$
7: **if** $x - n \geq 0$ **then**
8: mark all the rest of slots $(x - n)$ as occupied
9: return True
10: **else**
11: $N = N - X$
12: return False # *then continue with the others*
13: **end if**
14: **else**
15: return False
16: **end if**
17: **else**
18: **sort** $subtrees$ based on the β values eq. 1
19: **for** $subtree$ in $subtrees$ **do**
20: $done$ = MAPLEx(N,B,x,$subtree$)
21: **if** $done$ = True **then**
22: return $done$
23: **end if**
24: **end for**
25: **end if**

subtree (case III), the requested VMs will be divided into two groups. The aggregate bandwidth needed by the group being placed is determined as the minimum aggregated bandwidth between the two groups. As illustrated in algorithm 2 in lines 7–8, the head group has m VMs, and the tail group has $n - m$ VMs. The algorithm will only allocate $\min(m, n - m) \times B$ as the aggregated bandwidth needed by the VM group has a placement. However, differently than in Oktopus, MAPLE estimates the residual bandwidth of a server based on effective bandwidth. The variable $node.remainBW$ (line 3) in Alg. 1 is calculated by $(C_i - R^i_{eff})$, where C_i is the link capacity available at server i, and R^i_{eff} is the aggregated effective bandwidth (for the QoS target sought by the request under consideration) of the VMs already allocated to server i, as discussed in §III.

A. MAPLEx—Supporting Anti-Colocation Constraints

Anti-colocation constraints for VM placements have been studied previously [17], [39], [40], however the MAPLE algorithm as introduced above cannot accommodate them. In this section, we describe the MAPLEx algorithm, specified in Alg. 3, which enables us to use effective bandwidth to place VM ensemble requests involving anti-colocation constraints. The MAPLEx algorithm has a similar procedure to MAPLE, the essential difference bring in the inputs the algorithms take.

The input to the MAPLEx algorithm is a VM ensemble request $< N, B, x >$, where the new parameter x represents the

number of VM slots that will be occupied by a given request on each server. The introduction of x enables the algorithm to cope with different variations of VM location constraints that can be arbitrarily defined by the parameter x. In particular, two types of anti-colocation constraint can be supported:

- Setting $x = 1$ means that MAPLEx will be forced to allocate a maximum of one VM slot on each server to a given VM ensemble request. This equates to the anti-colocation constraint that *VMs from the same ensemble cannot be placed on the same server.*
- Setting $x = z$, where $z :=$ {the maximum number of slots on a server} means that MAPLEx will be forced to mark all the VM slots of a server occupied, once the server is allocated with a tenant's VM, regardless that the actually number of VM slots required is less than z. For example, a tenant requests 5 VMs ($|N| = 5$), while a given server has 10 slots ($x = z = 10$), then after placed with 5 tenant's VMs, the remaining 5 slots of that server will be marked as occupied, so the entire server is reserved to the tenant. This setup copes with the anti-colocation constraint that *the VMs from an ensemble cannot be placed on a server hosting VMs from other ensembles.*

The algorithmic structure of MAPLEx is similar to MAPLE—it proceeds in a similar manner. We thus explain only the MAPLEx specific aspects of Alg. 3. Line 5 creates the set of VMs X to be allocated to a given server. The size of X is the minimum number between n (size of N) and x, as we can see in lines 7–11, for the cases $x \geq n$, MAPLEx will assign the n VMs to the server and mark the remaining slots $(x - n)$ of the server as occupied; while $x < n$, MAPLEx allocates $(n - x)$ VMs to the server, reset $N = N - X$ (in line 11) and return false, subsequently the iterative process will pick up the updated N and continue to allocate N until the size of N equals to 0. The MAPLEx algorithm, as specified here, is generic in the sense that different forms of anti-colocation constraints can be specified by adjusting the x parameter in the input of VM ensemble request.

VI. Evaluation

A. Experimental Setup

For our experimental platform we used 4 Dell R720 servers and 1 physical switch to create an Emulated Datacenter (EDC). Each Dell server has 16 cores running at 2.6GHz, 128GB of RAM, and two 600GB hard disks. Each server is initially installed with a Ubuntu 12.04 system (the host OS), with the libvirt library [41] being used to manage up to 70 guest VMs deployed on each physical server. Fig. 6 presents our EDC, which is configured to emulate a simplified datacenter tree topology, without multiple paths between switches. The topology comprises virtual servers and virtual racks, as follows:

- *Virtual Servers (vServer)*—we group 5 VMs together to emulate a vServer, wherein all VMs are directly connected to a Virtual Switch (vSwitch). This vSwitch is accordingly denoted as vNIC (virtual Network Interface Card), functioning like a NIC of a server. Each vNIC is limited to have only 1 Gbit upstream and downstream bandwidth, to emulate the scenarios that 5 VMs share one bottleneck link within a server;
- *Virtual Rack (vRack)*—each of the 4 physical servers is used to emulate a rack of vServers. Within each physical server, there are 2 levels of vSwitches to connect 70 VMs: the top level vSwitch, denoted as Top-of-Rack (ToR) vSwitch, connects the low level vSwitches to the root physical switch; the low level vSwitches connect the VMs in groups.

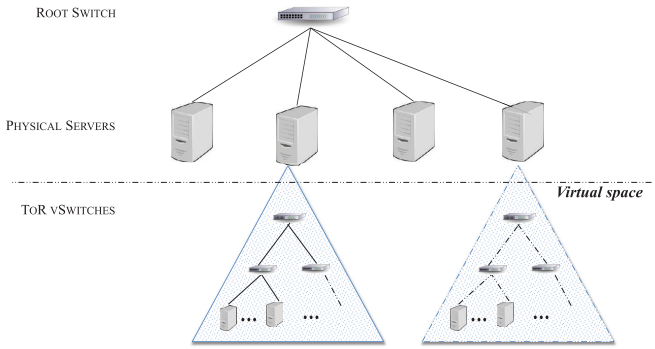


Fig. 6. The emulated datacenter has a single root switch connecting 4 physical servers. Each server hosts 70 VMs that are connected by 2 levels of vSwitches.

Three types of data-intensive applications are employed to run on the EDC: 1) For a data analysis application, we used a Hadoop application that runs word count on distributed documents; 2) for a data serving application, we use the combination of YCSB (data client) and Cassandra (data server), a benchmark recommended in cloudsuite [42]; finally, 3) for a data backup application, we created a program based on SCP utility, which simulates the scenario that a copy of data file is copied to a couple of remote nodes for backup purposes. To evaluate the efficacy of our emulation approach to building a test environment we preformed a preliminary investigation of the mean/maximum throughputs and run times of the three applications when deployed on the emulated EDC in comparison to an equivalent deployment on a real OpenStack$_{TM}$ based IaaS environment in a mixed used production/experimental datacenter in Waterford Institute of Technology. Results of this investigation, which are not presented here, confirm that the emulation approach provides a reasonable approximation of a production environment.

For VM ensemble requests, we created a program that randomly generates requests specified in a format $< N, B, T >$. It is similar to the hose model $< N, B >$ with an additional parameter T to indicate the type of application. $N \in [2, 10]$ is an integer value randomly drawn from a Gaussian distribution $\mathcal{N}(5, 1)$ with mean 5 and standard deviation 1. Because we use a mean of 5, each request will, on average, ask for 5 VM instances. In turn, we know that the emulated datacenter, having 280 VM slots, can more or less accept 56 requests. To simulate the dynamics of VM requests, each request is configured with a service time (selected randomly with an average value of 2100 s); when this elapses, the corresponding VMs will be removed from the EDC, some VM slots will therefore become available again. For each experimental run we generated at least 60 VM ensemble requests, the requests are

divided into 10 batches, and every 100 s one batch was fed to MAPLE/Oktopus. Given this, assuming the last request comes after 500 s a single experimental run lasted 2600 s. Results presented are based on 10 independent simulation runs.

We compared the performance of MAPLE with two variants of Oktopus [6]: Oktopus allocating network resources based on expected mean throughput, and Oktopus allocating network resources based on expected peak throughput. The mean throughput and peak throughput were the average values measured on the aggregated traffic traces collected at the initial experiments. After that, at each experimental run, all algorithms were given with the same sets of VM ensemble placement requests. Only the corresponding bandwidth requirements were substituted, according to the respective Oktopus variants. In our experiments for MAPLE/MAPLEx we configure EB Agents to start the process of generating new effective bandwidth estimates every $K = 300\,s$, with estimates being made on traces for intervals of $S = 50\,s$. Thus, when effective bandwidth for links change due to changes in VM placements on PMs, it can take up to 300 s for this to be recognized by the MAPLE controller.

The goal of this evaluation is to compare the involved VM placement techniques in two metrics: reject rates of VM ensemble requests and QoS violation rates. All QoS target violation rates were calculated offline based on the collected traffic traces, where in QoS targets are represented in the format $(delay, proportion)$. We use the following QoS targets: $(0.02\ s, 0.05)$, $(0.02\ s, 0.1)$, $(0.04\ s, 0.05)$, and $(0.2\ s, 0.1)$. We highlight that QoS delay requirements vary depending on the hosted applications, but delays of up to 200ms are generally considered acceptable for real-time interactive services [43] and private communications with an operator of a large enterprise datacenter supporting a range of transactional web-based services suggesting that a similar delay target is often acceptable.

B. Residual Bandwidth Required to Place New VMs

We start our study with an interesting question arose from our previous work [15], where the experiments showed that the use of effective bandwidth can result in a relatively optimal VM placement schemes, by which effective bandwidth is always an value between the involved applications' mean throughput and peak throughput. The question is "whether Oktopus produce a similar placement results by allocating bandwidths based on $\theta \times$ mean_rate, for $\theta > 1$?" The major issue of the described approach is that setting a fix value of θ cannot adjust to the dynamics of bandwidth requirements, thus resulting in bandwidth overprovisioning over time. The approach will only work unless there is a dedicated mechanism in place to dynamically reset θ in response to changes in network. In fact, the aggregated effective bandwidth needed by applications that does not grow linearly, e.g., the aggregated effective bandwidth of 3 flows is not equal to the sum of the three individual effective bandwidth of the flows, rather, it generally less than the sum. Thus, if we allocate the 3 flows with the bandwidth amount of $3 \times \theta \times$ mean_rate, that could result in over-provisioning.

Fig. 7 depicts the change in the level of additional bandwidth to be allocated when a new VM is placed on the server. This is calculated as $\frac{R_{eff}}{|\text{VMs}|}$, that is,the aggregated effective bandwidth, R_{eff}, divided by the number of allocated VMs, $|VMs|$. Here, we can observe the non-linear nature of effective bandwidth—reflecting the statistical multiplexing of traffic from numerous sources, which means that less incremental bandwidth needs to be provided to attain the same level of QoS. A detailed experimental and theoretical analysis regarding this nature of effective bandwidth can be can be found, respectively, in the work of Davy et al. [31] and Kelly [14].

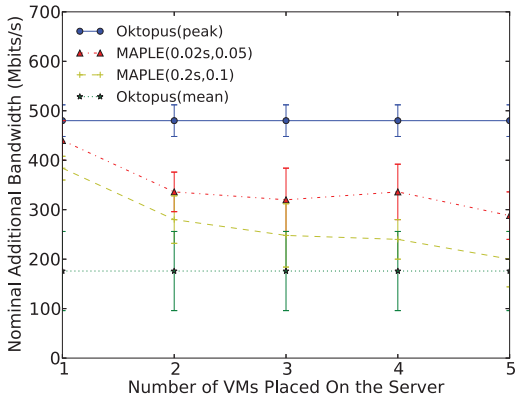


Fig. 7. Additional bandwidth that needs to be allocated when a new VM is placed on a server. MAPLE can place additional VMs based on the availability of less residual bandwidth than can Oktopus using peak throughput estimations, since it takes into account the statistical multiplexing effect captured by the effective bandwidth estimation.

C. Analysis of QoS Violations

We analyze QoS violations in the EDC that are associated with the operation of the VM placement algorithms. First, note that the QoS violation rate is calculated based on individual server's egress links. Namely, the violation rate is the proportion of a server's egress packets that experience delays longer than that specified in the QoS target. To examine the QoS violation at the overall datacenter scale, we employ two metrics: the *overall violation rate* as defined in Eqn. 2, and the *averaged violation rate* as defined in Eqn. 3. The *overall violation rate* is the sum of violation rates of all links divided by the total number of all observed links in the datacenter. However, this metric only shows the overall performance; it does not reflect the situation that the violations are very localized—most of the links have no, or tiny numbers, of QoS violation levels, while links that face QoS violations suffer frequent, significant violations. In this case, the averaged violation rate can indicate how strong the violations are on the links that suffer violations. It is important to have these complementary metrics, since both MAPLE and Oktopus algorithms attempt to allocate VMs in a same subtree with small residual bandwidths, in order to save routing costs, which will frequently result in scenarios where VMs are densely co-located around some links.

$$\text{overall\_violation\_rate} = \frac{\text{sum(QoS\_violation\_rates)}}{|\text{all\_links}|} \quad (2)$$

$$\text{averaged\_violation\_rate} = \frac{\text{sum(QoS\_violation\_rates)}}{|\text{links\_with\_violations}|} \quad (3)$$

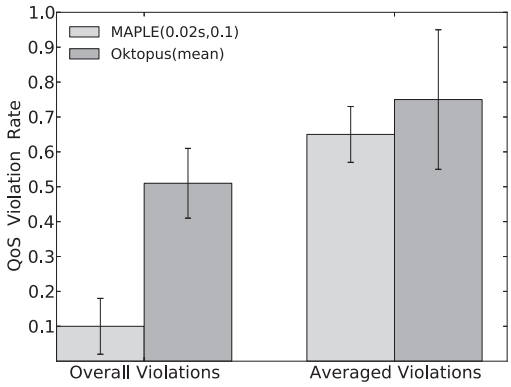


Fig. 8. QoS violation levels for QoS target (0.02 s, 0.1) for MAPLE and Oktopus. Oktopus over allocates network resources resulting in significant overall and localized levels of QoS violation. MAPLE's usage of effective bandwidth estimates means that QoS violations are within the acceptable range.

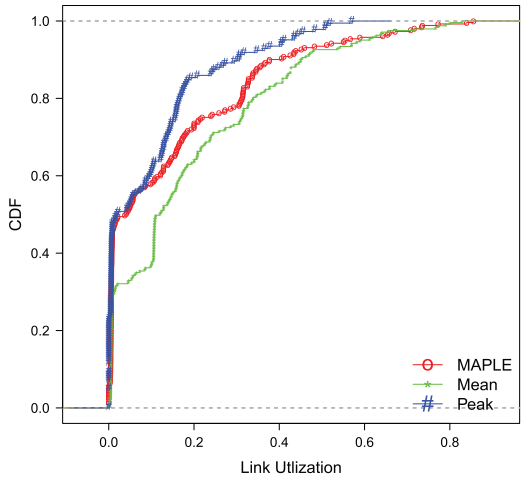


Fig. 9. CDF of the mean link utilization of all EDC links for which effective bandwidth estimates are generated, averaged over the full duration of an experimental run. We see that the majority of links are lightly loaded when MAPLE and both variants of Oktopus are used. Most QoS violations result from overloads of the smaller number of highly loaded links.

The evaluation of QoS target violation was calculated off-line based on the collected traffic traces. Given the scale of our EDC, it is not viable to collected and analyze traffic traces on every links for each run. Thus, we only focus on the links that have inter-vServer/inter-group traffic. Regarding the "all link" described in Eqn. 2, we mean all the links that in observations, namely the links with active traffic sniffers. Fig. 8 depicts QoS violations where the QoS target is $(0.02\ s, 0.1)$. The results compared the QoS violations incurred by two different sets of VM placements given by MAPLE and Oktopus when allocating based on mean throughout (denoted as Oktopus (mean) in the figure). Since VM placements given by Oktopus allocating using peak throughput generally cause no QoS violations or small violation rates, Fig. 8 only presents results for the other two algorithms.

From Fig. 8 we can observe that when allocating using mean throughputs to all VMs, Oktopus tends to have high probability ($\approx$60%) that packets will suffer with packet delays greater than 0.02 s. In particular, for the links where QoS violations actually occurs, we observe very strong violation rates, on average $\approx$77%. In contrast, MAPLE, which allocates on the basis of effective bandwidth estimates, generally keeps QoS violations within the target range, however, regarding the links with strong violations, MAPLE also suffers with strong violation rates, on average $>$65%. The CDF of mean link utilisation levels averaged across the duration of an experimental run presented in Fig. 9 indicates that the majority of violations result from overloads of a small number of highly loaded links.

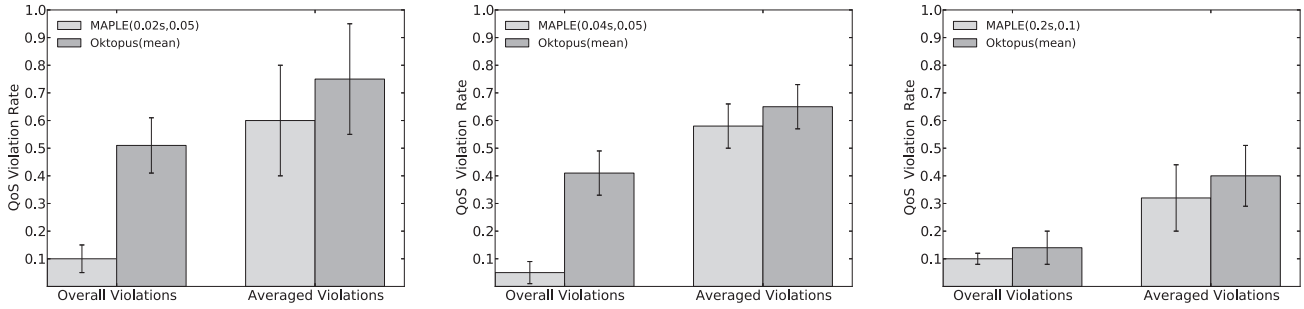
For a QoS sensitive network, it is also important to be able to control the level of QoS violation, being flexible to allocate just enough resources to maintain the QoS targets. Given three different QoS target $(0.02\ s, 0.05)$, $(0.04\ s, 0.05)$, and $(0.2\ s, 0.1)$, Fig. 10(b) and Fig. 10(c) show that MAPLE can keep QoS violation at acceptable levels for the targets (0.04 s, 0.05) and (0.2 s, 0.1), in terms of their *overall violation rates*. However, for the strictest QoS target, (0.02 s, 0.05), MAPLE did suffer with some congested links that lead to violation rates greater than 0.05, resulting in Fig. 10(a), where we see that the overall violation target is not met. Whilst MAPLE can meet the target in most cases, Oktopus results in varying violation rates, depending on the different delay targets—it never succeeds in keeping the rates within the acceptable range.

As we have seen in Fig. 8 and Fig. 10 that all the compared techniques have high *averaged violation rate*, as previously discussed low *overall violation rate* does not necessarily mean the *averaged violation rate* should be low, as in the cases described above for MAPLE. We investigated the traffic samples to find out the reason why *averaged violation rates* tend to be high, the summary of our finding is that this relates to a bursty application (Hadoop) running on a bursty network. As described by Chowdhury et al. [44] Hadoop traffic tends to be very bursty for some specific periods. Regardless of Hadoop, some SCP and YCSB traces also contribute some violation samples; we attribute this mainly to our Open vSwitch controlled network. Specially, during the initial configuration of the test-bed, we followed the open vSwitch community guidance [45] to set up the rate limit, where it is suggested to set a higher burst parameter for the vSwitch controlled network in order to manage bursty traffic, resulting in a bursty network and possibly congestion.

Although MAPLE occasionally fails to meet the QoS targets for some links that have strong violations, for most links ($>$80%) it meets the targets. Overall, it has much better performance than Oktopus using mean throughput regarding achieving QoS control.

D. Analysis of Rejection Rates of VM Ensemble Requests

Given that we know our test-bed generally can accommodate simultaneous placement of 56 requests, if the compared placement algorithms are provided with the number of requests that far beyond the corresponding EDC's capacity, this will result in large rejection rates. Therefore, in each run we only give 60 requests to a completely empty test-bed; We intentionally set the proportion of VM requests to be removed to be approximately 20% (i.e., $\approx$12 requests) in each experimental run, thus we give 18 requests to the test-bed in a state that some of its requests and the corresponding VMs have been removed.



(a) QoS target (0.02s, 0.05) (b) QoS target (0.04s, 0.05) (c) QoS target (0.2s, 0.1)

Fig. 10. QoS violation rates for different delay based QoS targets. MAPLE mostly can keep violations below the specified target level or, if not, at a relatively low level, whereas Oktopus (mean) never does.

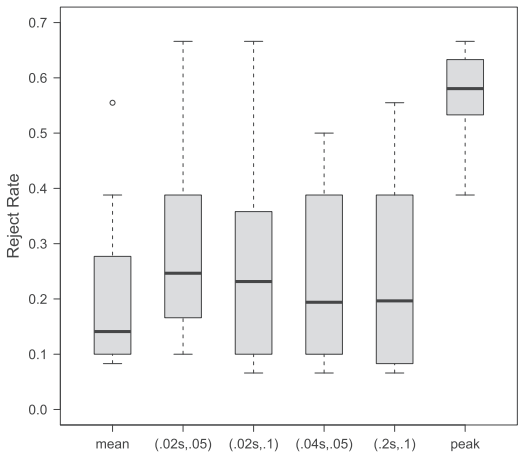


Fig. 11. Oktopus (mean) has the lowest rejection rate, while MAPLE's rejection rates are comparable to it, but dependent on the QoS targets. Oktopus (peak) has the highest rejection rate.

In each run, the EDC was populated with VMs until it was almost fully loaded, that is, when it could not accept new VM ensemble placement requests any more. Fig. 11 depicts the reject rates for MAPLE with four different QoS targets and Oktopus using mean and peak throughput. We observe that the peak throughput approach has a significantly higher reject rates, compared to the other algorithms. The first two members of MAPLE's family seem to have similar reject rates, which is somehow reasonable as they have the same delay limit. In particular, when many flows are involved, the aggregated effective bandwidth with similar QoS targets tends to converge. With a less strict QoS requirement (0.04 s, 0.05), MAPLE produces a comparable rejection rate to Oktopus (mean), which has the lowest reject rate (at the expense of QoS violations for reasonable QoS targets).

Generally, given a less strict QoS requirement, MAPLE should produce lower rejection rates. However, from the box plot in Fig. 11, we observe that when comparing the maximum reject rates between MAPLE(0.04 s, 0.05) and MAPLE(0.2 s, 0.1), the latter shows a higher value. We investigated the related batches of ensemble requests, and found that is due to MAPLE (0.2 s, 0.1) was able to accept some requests requesting bigger number of VMs, resulting in subsequently rejecting more requests of smaller number of VMs; whereas MAPLE(0.04 s, 0.05) rejected one or two requests of big number of VMs thus was able to accept more requests of small number of VMs.

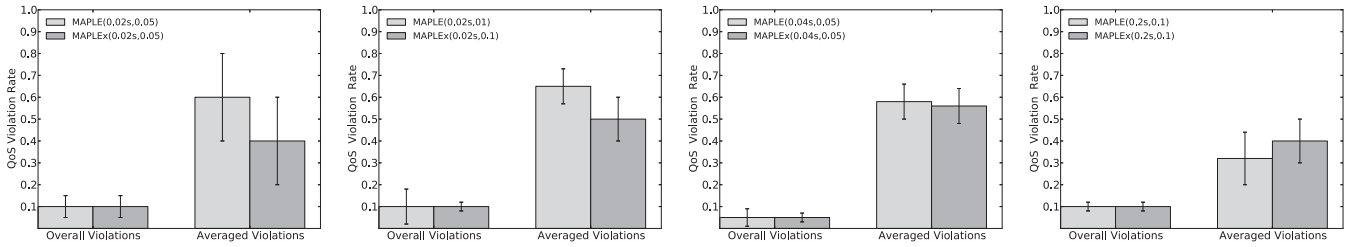
Overall the rejection rates produced by MAPLE are in between the rejection rates of Oktopus using mean and Oktopus using peak. The values of rejection rates of MAPLE seems more varied, which reflects the dynamics of effective bandwidth requirements needed by the datacenter applications, so that the amount of VMs can be placed is varied. Oktopus variants that always reserve the same bandwidth tend to be more static in terms of rejection rates.

Based on the experimental results on QoS violation rates and request reject rates, overall we can conclude that MAPLE is relatively resource-conserving while providing targeted QoS statistical guarantees.

E. Analysis of MAPLEx

To evaluate MAPLEx it is given the same batches of requests that were given to Oktopus and MAPLE in the experiments described above. However, in these batches of requests, we randomly select approximately 70% of the requests to have the anti-colocation constraints, such that 30% of the overall requests have the constraint that VMs from the same ensemble cannot be placed on the same server and 40% of the overall requests have the constraint that the VMs from an ensemble cannot be placed on a server hosting VMs from other ensembles.

Fig. 12 compares the QoS violation rates incurred by the placements of MAPLE and MAPLEx for this request pattern. We see that, since placement decisions are based on available effective bandwidth, MAPLEx, like MAPLE, is typically able to control the delay violation rates to meet the given targets. However, as explained earlier, due to the presence of bursty applications, there also exists some link congestion, leading to some undesired QoS violations. For three of the four QoS targets investigated, MAPLEx has a better averaged violation rate, which captures the degree to which violations occur when they do occur. This is due to MAPLEx tending to distribute VMs across more PMs than does MAPLE, due to the presence of the anti-colocation requests (and, if no PMs are available to reject more placement requests). Thus, those links that do suffer violations due to overload generally have lower mean carried load for the MAPLEx case in comparison to the MAPLE case, so



(a) QoS target (0.02s, 0.05) (b) QoS target (0.02s, 0.1) (c) QoS target (0.04s, 0.05) (d) QoS target(0.2s, 0.1)

Fig. 12. QoS violation rates for different delay based QoS targets. In most cases MAPLE and MAPLEx can keep violations below the specified target level, although some links suffer relatively high violation rates.

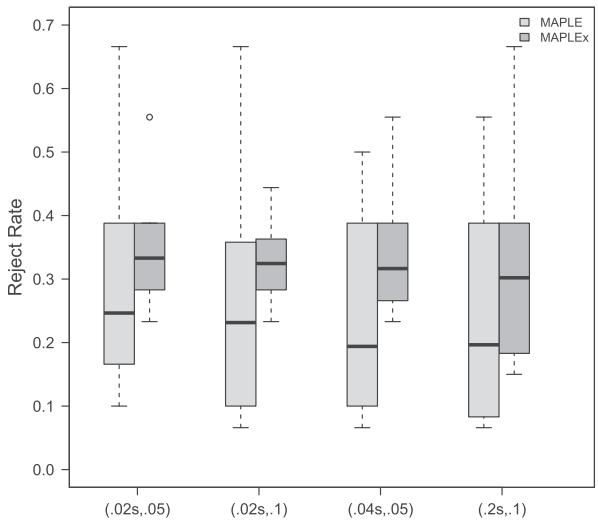


Fig. 13. Rates of ensemble placement request rejection—MAPLEx tends to show higher placement rejection rates in comparison to MAPLE. The presence of anti-colocation constraints, particularly those mandating that only VMs from a given ensemble can be placed on a given server effectively reduce the overall capacity of the cluster to accommodate VM ensembles.

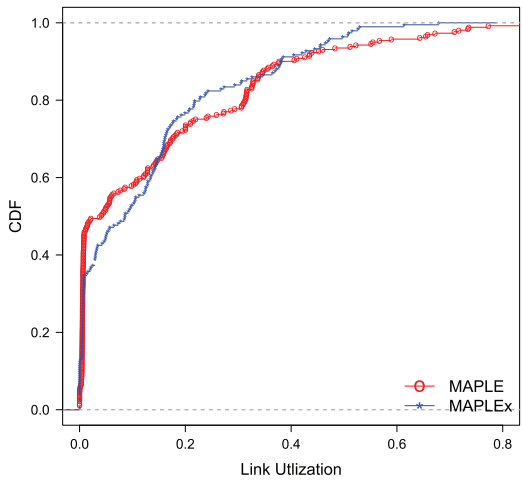


Fig. 14. CDF of the mean link utilization of all EDC links for which effective bandwidth estimates are generated, averaged over the full duration of an experimental run for MAPLE and MAPLEx. We see that for MAPLEx, links have generally lower mean utilization, resulting from MAPLEx tending to place VMs across more PMs (to satisfy anti-colocation constraints) and its tendency to reject more VM ensemble placement requests than does MAPLE.

that the impact of transient high traffic demands from VMs is lessened in terms of the duration of the overload, leading to a lower level of QoS violation.

Fig. 13 compares the request rejection rates; we can observe that anti-colocation constraints supported in MAPLEx frequently result in more requested placements being rejected. In particular, the constraints that VM ensemble does not share server would cause some slots marked occupied but not actually being used—effectively lowering the capacity of the cluster to accommodate VM ensembles. Fig. 14 shows that with MAPLEx links generally have lower mean utilization than with MAPLE. This results from MAPLEx tending to place VMs across more PMs (to satisfy anti-colocation constraints) and its tendency to reject more VM ensemble placement requests (because there is less available PM capacity due to some PMs hosting VMs that cannot share resources with VMs from other ensembles).

VII. Conclusion

In this article we described our use of a test-bed that utilizes four physical servers to emulate a datacenter cluster having 280 VM slots running three types of data-intensive applications. Using this larger scale of test-bed running different applications, we confirmed the findings of our previous work [15] that the Oktopus variant, which allocates bandwidths to tenant VMs based on mean throughput, tends to suffer with low QoS performance, whereas the Oktopus variant based on allocating peak throughput tends to waste resources.

The optimal amount of bandwidth for provisioning should lie between the mean throughput and peak throughput, however existing network-aware VM placement schemes do not determine the optimal value for the bandwidth allocation. They are designed to maximize throughput of datacenter networks, but not to deliver predictable performance in terms of the latency experienced by users of hosted applications. Our proposed system, MAPLE, provides this form of predictability by placing VMs in a tenant application's VM ensemble based on ensuring that there is sufficient effective bandwidth available between all pairs of VMs in the ensemble when they are placed on datacenter servers. Experimental results based on the test-bed have shown that MAPLE typically succeeds in meeting QoS targets whilst simultaneously allocating both computing and network resources in an efficient manner.

By default, MAPLE tends to co-locate the VMs from same ensembles. To cope with the anti-colocation requirements that

are often seen in application VM ensemble placement requests, we describe MAPLEx, a MAPLE extension that accommodates anti-colocation constraints, which in high resource demand scenarios can lead to overprovisioning of server capacity. Our experiments show that, like MAPLE, MAPLEx was able to provide statistical QoS guarantees through its use of effective bandwidth estimations.

Future work will explore different approaches for selecting subsets of servers as candidates for placement of incoming VM ensemble requests. We also plan to extend MAPLE to support elastic VM ensembles in which constituent VMs can be instantiated and/or re-provisioned on-the-fly to handle growing application demands. One limitation of MAPLE is that the VM placements it makes, whilst being appropriate at the time the placements are made, may lead to sub-optimal placements over time, especially if VM ensembles are deployed for extended time periods. We plan to explore how MAPLE could be extended to periodically identify VM migrations that would result in more optimal global placements. We also intend to explore the use of effective bandwidth estimation in the placement of more complex ensembles which include (virtual) network appliances (for example, load balancers and intrusion detection systems) as well as application VMs.

REFERENCES

[1] A. Greenberg *et al.*, "Vl2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM*, 2009, pp. 51–62.

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

[3] A. Shieh, S. Kandula, A. Greenberg, and C. Kim,"Seawall: Performance isolation for cloud datacenter networks," in *Proc. HotCloud*, 2010, pp. 1–7.

[4] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *Proc. WIOV*, 2011, pp. 1–8.

[5] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM*, 2010, pp. 1154–1162.

[6] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. ACM SIGCOMM*, 2011, pp. 242–253.

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

[8] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: Incorporating time-varying network reservations in data centers," in *Proc. ACM SIGCOMM*, 2012, pp. 199–210.

[9] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing," in *Proc. ACM SIGCOMM*, 2013, pp. 351–362.

[10] K. LaCurts, D. Shuo, A. Goyal, and H. Balakrishnan, "Choreo: Network-aware task placement for cloud applications," in *Proc. ACM Internet Meas. Conf. (IMC)*, 2013, pp. 191–204.

[11] F. Wuhib, R. Yanggratoke, and R. Stadler, "Allocating compute and network resources under management objectives in large-scale clouds," *J. Netw. Syst. Manage.*, vol. 23, no. 1, pp. 111–136, 2015.

[12] Amazon Inc. *Amazon EC2 Instances* [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[13] Rackspace Inc. *Rackspace Cloud* [Online]. Available: http://www.rackspace.com/

[14] F. Kelly, "Notes on effective bandwidths," *Stochastic Netw.: Theory Appl.*, vol. 4, pp. 141–168, 1996.

[15] R. Wang, R. Esteves, L. Shi, J. Wickboldt, B. Jennings, and L. Granville, "Network-aware placement of virtual machine ensembles using effective bandwidth estimation," in *Proc. IFIP/IEEE 10th Int. Conf. Netw. Serv. Manage. (CNSM)*, 2014, pp. 100–108.

[16] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *Proc. IEEE INFOCOM*, 2012, pp. 2861–2865.

[17] L. Shi, B. Butler, D. Botvich, and B. Jennings, "Provisioning of requests for virtual machine sets with placement constraints in IaaS clouds," in *Proc. IFIP/IEEE Int. Conf. Integr. Netw. Manage. (IM'13)*, 2013, pp. 499–505.

[18] D. Breitgand and A. Epstein, "SLA-aware placement of multi-virtual machine elastic services in compute clouds," in *Proc. IFIP/IEEE 12th Symp. Integr. Netw. Manage. (IM'11)*, 2011, pp. 161–168.

[19] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," in *Proc. 18th ACM Conf. Comput. Commun. Secu. (CCS'11)*, 2011, pp. 401–412.

[20] D. Perez-Botero, J. Szefer, and R. B. Lee, "Characterizing hypervisor vulnerabilities in cloud computing servers," in *Proc. Int. Workshop Secur. Cloud Comput. (Cloud Comput.'13)*, 2013, pp. 3–10.

[21] M. F. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Commun. Surv. Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart. 2012.

[22] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, pp. 567–619, 2015.

[23] C. Guo *et al.*, "Secondnet: A data center network virtualization architecture with bandwidth guarantees," in *Proc. ACM CONEXT*, 2010, pp. 15:1–15:12.

[24] R. Kanagavelu, B.-S. Lee, N. T. D. Le, L. N. Mingjie, and K. M. M. Aung, "Virtual machine placement with two-path traffic routing for reduced congestion in data center networks," *Comput. Commun.*, vol. 53, pp. 1–12, Nov. 2014.

[25] S. Angel, H. Ballani, T. Karagiannis, G. O'Shea, and E. Thereska, "End-to-end performance isolation through virtual datacenters," in *Proc. 11th USENIX Conf. Oper. Syst. Des. Implement.*, 2014, pp. 233–248.

[26] O. Biran *et al.*, "A stable network-aware VM placement for cloud systems," in *Proc. 12th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGrid'12)*, 2012, pp. 498–506.

[27] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. IEEE INFOCOM*, 2011, pp. 71–75.

[28] T. Duong-Ba, T. Nguyen, B. Bose, and T. Tran, "Joint virtual machine placement and migration scheme for datacenters," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2014, pp. 2320–2325.

[29] C. Courcoubetis, V. A. Siris, and G. D. Stamoulis, "Application and evaluation of large deviation techniques for traffic engineering in broadband networks," in *Proc. ACM SIGMETRICS*, 1998, pp. 212–221.

[30] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 7, pp. 968–981, Sep. 1991.

[31] A. Davy, D. Botvich, and B. Jennings, "Revenue optimized IPTV admission control using empirical effective bandwidth estimation," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 599–611, Sep. 2008.

[32] A. Davy, D. Botvich, and B. Jennings, "An efficient process for estimation of network demand for QoS-aware IP network planning," in *Proc. IEEE 6th Workshop IP Oper. Manage. (IPOM'06)*, Oct. 2006, pp. 120–131.

[33] VMware Inc. (2014). *VMware vCenter* [Online]. Available: http://www.vmware.com/products/vcenter-server

[34] N. R. Mysore *et al.*, "Portland: A scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM*, 2009, pp. 39–50.

[35] G. Combs. (2006). Wireshark Hompage [Online]. Available: http://www.wireshark.org/

[36] L. Shi, J. Furlong, and R. Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers," in *Proc. IEEE Manage. Cloud Syst. Workshop (MoCS)*, 2013, pp. 9–15.

[37] M. Sheikhalishahi, R. M. Wallace, L. Grandinetti, J. L. Vazquez-Poletti, and F. Guerriero, "A multi-capacity queuing mechanism in multi-dimensional resource scheduling," in *Adaptive Resource Management and Scheduling for Cloud Computing*, F. Pop and M. Potop-Butucaru, Eds. New York, NY, USA: Springer, Jul. 2014, no. 8907, pp. 9–25.

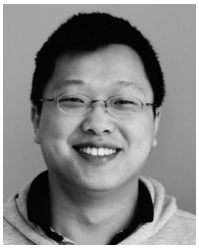
[38] N. Bitar, S. Gringeri, and T. Xia, "Technologies and protocols for data center and cloud networking," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 24–31, Sep. 2013.

[39] E. Bin *et al.*, "Guaranteeing high availability goals for virtual machine placement," in *Proc. IEEE 31st Int. Conf. Distrib. Comput. Syst. (ICDCS'11)*, 2011, pp. 700–709.

[40] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving performance and availability of services hosted on IaaS clouds with structural constraint–Aware virtual machine placement," in *Proc. IEEE Int. Conf. Serv. Comput. (SCC'11)*, 2011, pp. 72–79.

[41] Libvirt.org. *Libvirt API* [Online]. Available: http://libvirt.org/

[42] M. Ferdman *et al.*, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," in *Proc. ASPLOS*, 2012, pp. 37–48.
[43] Cisco Inc. *Cisco Online Document* [Online]. Available: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/5125-delay-details.html
[44] M. Chowdhury, M. Zaharia, J. Ma, M. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM*, 2011, pp. 98–109.
[45] Openvswitch Community. *Rate-Limiting VM Traffic Using QoS Policing* [Online]. Available: http://openvswitch.org/support/config-cookbooks/qos-rate-limiting/



Runxin Wang (M'13) received the B.Sc. degree from Guangzhou University, Guangzhou, China, in 2006, and the M.Sc. degree from the National University of Ireland, Maynooth, Ireland, in 2008. He is currently pursuing the Ph.D. degree with the Telecommunications Software and Systems Group (TSSG), Waterford Institute of Technology, Waterford, Ireland. He was an Intern at Sun Microsystems, Ireland, between 2007 and 2008. Later he joined TSSG and conducted research in some funded data mining projects. He started the Ph.D. study in the area of network management in 2012, with the focus on applying data mining and traffic analysis techniques for network management. In 2014, he spent a month as a Visiting Researcher at the Federal University of Rio Grande do Sul, Brazil. He is currently on secondment at EMC^2 Research Europe in Cork, Ireland, where he is investigating the use of Software Defined Networking Technologies in datacenters.

Juliano Araujo Wickboldt (M'09) received the B.Sc. degree in computer science from Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil, in 2006, and the M.Sc. degree from UFRGS (conducted in a joint project with HP Labs Bristol and Palo Alto). He is currently pursuing the Ph.D. degree at the Federal University of Rio Grande do Sul (UFRGS). He was an Intern at NEC Labs Europe in Heidelberg, Germany for one year between 2011 and 2012. Between 2013 and 2014, he was a substitute Professor with UFRGS. In the beginning of 2015, he was in Waterford, Ireland visiting the Telecommunications Software and Systems Group (TSSG) of the Waterford Institute of Technology, working within the Emerging Networks Laboratory. His research interests include cloud resource management and software-defined networking.

Rafael Pereira Esteves (M'09) received the B.Sc. and M.Sc. degrees, both in computer science, from the Department of Informatics of the Federal University of Pará (UFPA), Tucuru, Brazil, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from the Institute of Informatics (INF), Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 2014. During his Ph.D., he was a Visiting Student at the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada. Currently, he is a Professor with the Federal Institute of Rio Grande do Sul (IFRS), Bento Gonçalves, Brazil. His research interests include network and service management, network virtualization, cloud computing, and software-defined networking.

Lei Shi (M'09) received the master's degree in computer science from Uppsala University, Uppsala, Sweden, in 2006; and the Ph.D. degree from the Faculty of Mathematics and Computer Science, University of Göttingen, Göttingen, Germany, in 2010. In 2013, he was a Visiting Professor in Technion, Israel. Currently, he is a Research Fellow with the Telecommunications Software and Systems Group (TSSG), Waterford Institute of Technology, Waterford, Ireland and a Visiting Researcher at the University of Massachusetts Amherst, Amherst, MA, USA. His research interests include network monitoring, data centre resource management, and mobile cloud computing.

Brendan Jennings (M'05) received the B.Eng. and Ph.D. degrees from Dublin City University, Dublin, Ireland, in 1993 and 2001, respectively. He is the Head of Graduate Studies for Waterford Institute of Technology, Waterford, Ireland, where he is also active as a Senior Researcher within the Emerging Networks Laboratory, Telecommunications Software, and Systems Group. Within the past three years, he has spent periods as a Visiting Researcher at KTH—Royal Institute of Technology, Stockholm, Sweden, and in EMC^2 Research Europe, Cork, Ireland. He regularly serves on the organization and technical program committees of a number of network and service management related conferences. His research interests include network management, cloud computing, and nanoscale communications.

Lisandro Zambenedetti Granville (M'03) received the M.Sc. and Ph.D. degrees, both in computer science, from UFRGS in 1998 and 2001, respectively. He is an Associate Professor with the Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil. He has served as a TPC Co-Chair of IFIP/IEEE DSOM 2007, IFIP/IEEE NOMS 2010, TPC Vice-Chair of CNSM 2010, and the General Co-Chair CNSM 2014. He is also Chair of the IEEE Communications Society's Committee on Network Operations and Management (CNOM) and Co-Chair of the Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF). He is the Director of the Research and Development Center for Digital Technologies for Information and Communication (CTIC) and Vice-President of the Brazilian Computer Society (SBC). His research interests include management of virtualization for the future internet, software-defined networking (SDN) and network functions virtualization (NFV), and P2P-based services and applications.