Network-aware Placement of Virtual Machine Ensembles using Effective Bandwidth Estimation

Runxin Wang*, Rafael Esteves‡, Lei Shi*, Juliano Araujo Wickboldt‡,
Brendan Jennings*, Lisandro Zambenedetti Granville‡
*TSSG, Waterford Institute of Technology, Ireland
‡Federal University of Rio Grande do Sul, Brazil
Email: {rwang, lshi}@tssg.org, bjennings@ieee.org, {rpesteves, jwickboldt, granville}@inf.ufrgs.br

***Abstract*—Modern datacenters rely heavily on virtualization technologies to offer customized computing and network resources on demand to a large number of tenant applications. However, efficiency in resource utilization delivered by virtualization technologies that exploit statistical multiplexing of resources across applications means that predictability in performance remains a challenge. Allocation of network bandwidth is particularly difficult, given the variability of traffic flows between the components of multi-tier applications. 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 scheme that uses empirical estimations of the effective bandwidth required between servers to ensure that QoS violations are within targets specified in the SLA for the tenant application. Experimental results obtained using traffic traces collected from an emulated datacenter show that, in contrast to the Oktopus network-aware VM placement system, MAPLE is able to allocate computing and network resources in a manner that balances efficiency of resource utilization with performance predictability.**

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 users. 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 the unpredictable network performance in datacenters had become a bottleneck to many cloud-hosted applications, with several techniques having been developed to address this issue [3]–[10].

The 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 [11] and Rackspace [12] 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. Overprovisioning is a pragmatic approach given that it is extremely difficult for tenants to accurately predict inter-VM bandwidth requirements for their VM ensembles in advance. Thus, tenants can be provided with weak statistical guarantees that QoS targets in their 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 the issues with overprovisioning 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]–[10]. Typically, 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 that 2% of packets should be delayed by more than 50ms."

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 the servers is sufficient. Kelly [13] defined effective bandwidth as the "minimum amount of bandwidth required by a traffic source to maintain specified QoS targets." We do not require either the *a priori* reservation of bandwidth, or 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 of empirically computed estimates of effective bandwidth on the network paths in the admission decision to ensure that the likelihood of SLA violations is minimal. This approach allows the datacenter provider to specify QoS targets for hosted applications in terms of delay, ensuring that network resources are utilized efficiently.

In this paper, we describe the MAPLE system, which has been developed to accomplish the following objectives: 1) provision of predictable network performance to tenant VM ensembles, 2) optimal joint-allocation of network and computing resources, and 3) satisfaction of application QoS

targets. The design of MAPLE needs to address the challenges in applying effective bandwidth techniques to manage datacenter networks. First, we design a network-aware VM placement algorithm that can utilize the effective bandwidth technique to produce optimal VM placement solutions in a timely manner. Second, we seek to de-centralize the effective bandwidth estimations to improve the run-time performance. Given these considerations, MAPLE comprises two functional modules: a centralized resource allocation manager that controls VM placement for a server cluster and per-server effective bandwidth measurement agents.

The paper is structured as follows. §II presents the related work on network allocation systems in datacenters. The concept of effective bandwidth is introduced in §III. §IV and §V describe the MAPLE system design and the network-aware VM placement algorithm respectively. The experimental evaluation is presented in §VI, where MAPLE is compared to Oktopus [6], a well known system described in the literature. The paper concludes in §VII, where topics for future work are briefly outlined.

II. Related Work

A number of recent publications have focused on network resource allocation in datacenters. However, to the best of our knowledge, 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 overview of the literature, we refer to Bari *et al.* [14] as well as Jennings and Stadler [15].

Guo *et al.* [16] 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 guarantee; this moves the bandwidth reservation tasks from switches to hypervisors.

Ballani *et al.* [6] proposed Oktopus, in which VM ensembles are placed based on virtual cluster abstractions. In their work, all VMs are modelled 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 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 either be the expected mean bandwidth of the traffic generated by the VM or the expected peak bandwidth.

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.

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. The work presented in [5], [17], [18] also addresses modelling 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. Preliminaries

A. Effective Bandwidth and its Estimation

Effective bandwidth is the minimum amount of bandwidth required by a traffic source (e.g., 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 on the whole system, including link capacity, traffic characteristics, and the QoS target [19]. For example, if a link has capacity of 1 Gbps, given 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 milliseconds. The question that arises is whether the given link can accommodate the 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 milliseconds. However, the probability of this situation actually happening is 1% (assuming VMs are independent), therefore the QoS target is not violated, and the link can accommodate the VMs. In order 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 as the chance that they both reach peak is small enough to statistically guarantee that the QoS target is not violated. This discussion indicates that the effective bandwidth lies somewhere between the source's mean throughput and peak throughput [13].

Effective bandwidth can be estimated analytically through the application of large deviation theory [19]. However, in MAPLE, we employ an empirical approach based on 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, is compared to the available bandwidth to assess whether the VM can be placed on that server. The effective bandwidth estimation technique is taken from Davy *et al.* [20] and is summarised as follows. Let $delay_{max}$ be the nominal maximum delay and let p_{delay} be the percentage of traffic which 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 which 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, it employs 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 is provided in [20]).

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 the VMs placed on the server. If a server's outgoing link capacity is C, with N VMs placed on it, each allocated 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 the system.

In contrast, in MAPLE 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 1000 Mbps. Static reservation refers to the method that reserves maximum throughput (940 Mbps) for the VMs and computes the residual bandwidth accordingly; whereas

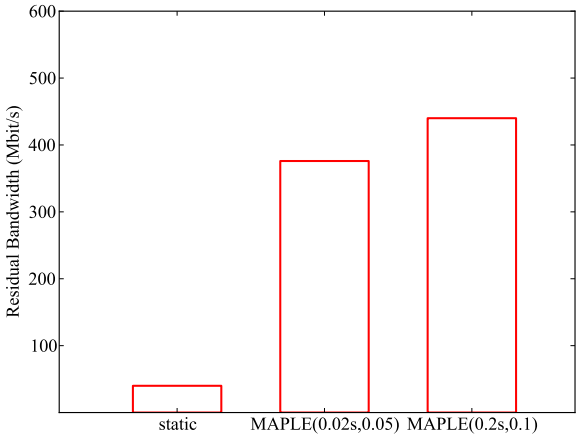


Fig. 1: The residual bandwidths of a server that has link capacity of 1 Gbps. When peak bandwidths are reserved for each VM little residual bandwidth remains. In contrast, when bandwidth is allocated based on effective bandwidth the available residual bandwidth is significantly higher, with the value depending on the stringency of the QoS target.

effective adjustment refers to adjusting the residual bandwidth based on the effective bandwidth estimates. In this case the latter determines that an aggregated bandwidth of 640 Mbps is sufficient for the two VMs for achieving a given QoS target (0.02s, 0.05)—no more than 5% of packets suffer delay longer than 0.02s; whereas given a lower QoS target (0.1s, 0.2) the residual bandwidth is higher.

IV. MAPLE System Architecture

MAPLE is designed to manage the joint allocation of network and computing resources in datacenter clusters in order 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. 2, these are: 1) the MAPLE Controller, which is deployed on a cluster management server and interacts with a cluster manager such as VMware vCenter [21]; and 2) effective bandwidth estimation agents (EB Agents) deployed on each server, which analyse 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 installed on every server on 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 analyser [23]). The traces are then used to estimate effective bandwidth using the approach outlined in §III. EB Agents collect traces at K minute intervals, each time storing S minute long traces. 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

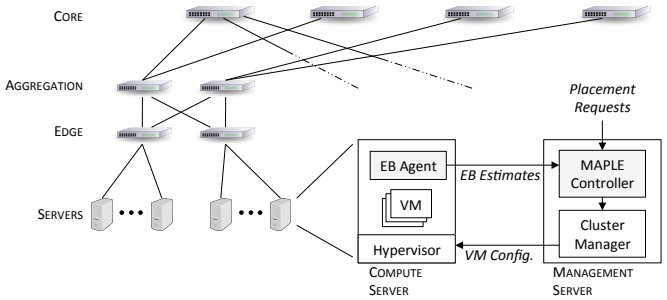


Fig. 2: The MAPLE system architecture. MAPLE is comprised of Effective Bandwidth agents (EB Agents) residing in each server and a MAPLE Controller residing in a management server. EB Agents send effective bandwidth estimates upon request to the MAPLE Controller. The MAPLE Controller processes VM placement 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 [22]), MAPLE is agnostic of the topology since it collects bandwidth estimates at servers only.

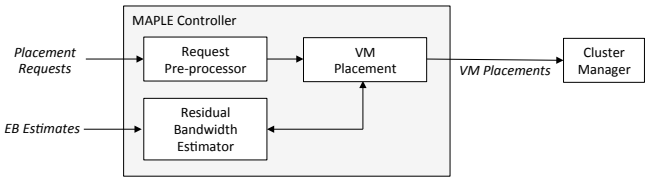


Fig. 3: The MAPLE Controller. Incoming requests for VM ensemble placement are first pre-processed to account for VMs that need to be placed together. VM placement decisions are then made taking into account the estimated available residual bandwidth at each server. Identified placements are passed to a Cluster Manager for configuration.

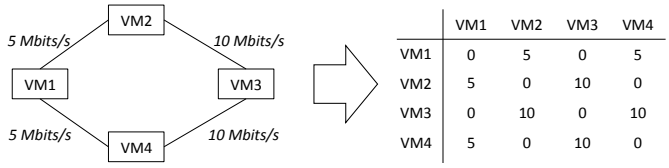
from the last five stored estimates. As all of the trace analysis is done locally on the server there is minimal overhead incurred in EB Agent to MAPLE Controller communications.

B. MAPLE Controller

The MAPLE controller handles incoming requests for VM ensemble placement, deciding if the request can be accepted and computing the placement if it can. As depicted in Fig. 3 it is comprised of three functional entities, which we now describe.

1) Request Pre-processor: The Request Pre-processor receives VM ensemble placement requests which 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. 4 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 Pre-processor also performs some preprocessing aimed to optimise the subsequent VM placement. It allows



	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. 4: A tenant VM ensemble topology specification indicating 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 pre-processor in the MAPLE Controller groups these VMs and generates a simplified traffic matrix.

placement requests to specify that particular VM sets should be placed together on the same server given the expectation that they will interact heavily with each other. Given this, the pre-processor and MAPLE Controller treat such VMs as a single VM, and simplify the traffic matrix accordingly.

2) Residual Bandwidth Estimator: This entity manages and queries the 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 inform it of 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 purposes. 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 pre-processed 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 network-aware VM placement algorithm specified in §V. If the VM ensemble can be safely placed the placement details are passed to the Cluster Manager which applies configurations on the servers accordingly.

V. Network-aware VM Placement Algorithm

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], [17], [18], 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 MAPLE algorithm applies the First Fit Decreasing (FFD) approach to search for VM placement solutions—FFD has been widely applied to VM placement problems, (see, *inter alia*, [17], [18], [24]). The algorithm first sorts the servers in decreasing order of their β values using Eqn. 1 and

commences searching. The approach is known as "first fit" as it stops searching once it founds 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 decreasing 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 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 decreasing order of residual bandwidth. The coefficient $g \in [0, 1)$ is applied to reduce the influence of residual VM level. 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 residual bandwidths, those with fewer remaining VM slots will be searched first.

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

Note 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 VM in request and B is the set of associated expected peak bandwidth utilization. To simplify the presentation of the algorithm and without loss of generality, we assume here 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], [22], [25]. When there are n requests arriving at a time t, they 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 a same server;
2) Case II, addressed in lines 10-16 – If a node has subtrees, the subtrees will be sorted decreasingly based on their β values, and MAPLE attempts to allocate the entire VM ensemble placement request into the same subtree. Note 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.

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 eq.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 $tail$

In an approach similar to that taken by Oktopus [6], whenever a VM request cannot be entirely placed into one subtree (case III), the requesting VMs will be divided into

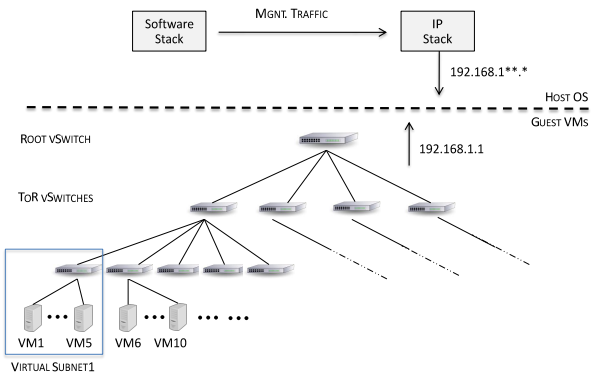


Fig. 5: The virtualized datacenter has a single root vSwitch and 20 virtual subnets distributed into 3 layers, each virtual subnet comprising 5 VMs and 1 vSwitch, simulating the scenario of 5 VMs competing a server's bottleneck link. The routing table in the host OS is rewritten to forward the packets of the specific IP addresses (192.168.1**.*) into the corresponding guest VMs.

two groups. The aggregate bandwidths needed by each group is determined as the minimum aggregated bandwidths 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 bandwidths needed by each VM group, respectively. However, unlike 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.

VI. EVALUATION

A. Experimental Setup

We emulated a virtual datacenter network using a single Dell R720 machine with 16 cores running at 2.6GHz, 128GB of RAM and two 600GB hard disks. Within this machine we instantiated a virtualized datacenter, as depicted in Fig. 5, comprising VMs (each having a statically assigned IP address) and virtual switches, all configured to emulate a simplified datacenter tree topology (without multiple paths between switches). In the arrangement VMs are grouped together in groups of 5 within a virtual subnet that has 1 Gbps data rates for both upstream and downstream traffic, thus emulating the scenarios of 5 VMs within a server that compete for the server's link capacity of 1 Gbps. We instantiate 20 such virtual subnets. Every 5 subnets is connected to one virtual switch (denoted as vSwitch in Fig. 5) with 10 Gbps upstream and downstream capacity. There are 4 virtual switches connecting subnets and one root virtual switch connecting those 4 virtual switches. The machine is initially installed with a Ubuntu 12.04 system (the host OS), with the libvirt library being used to manage the 100 guest VMs.

To emulate bulk data transfers within a datacenter network, we created a program (installed on host OS) that randomly asks multiple VMs to simultaneously upload, via the SCP utility, a dataset of size 300MB to a specified set of other VMs. For the tenant requests, we created a program that randomly generate tenant requests specified in the hose model $< N, B >$, where $N \in [2, 10]$ 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 the emulated datacenter which has 100 VM slots can more or less accept 20 requests, and in each experimental run we generate 30 requests to ensure that the datacenter will be (almost) fully loaded.

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 (176 Mbps) and peak throughput (480 Mbps) were the average values measured on the SCP traffic traces collected at the initial experiments. After that, at each experimental run, all the algorithms were given with the same sets of VM ensemble placement requests; only the corresponding bandwidth requirements were changed to (176 Mbps, 480 Mbps) according to the respective Oktopus algorithms. We have 20 runs of experiments, along which we sampled traffic traces summed up to $\approx$ 100 GB. QoS target violation rates were calculated offline, using these collected traffic traces.

In the following analysis of experimental results QoS targets are represented in the format $(delay, proportion)$—we use the following QoS targets: $(0.02s, 0.05)$, $(0.02s, 0.1)$, $(0.04s, 0.05)$, and $(0.2s, 0.1)$. We note that QoS delay requirements vary depending on the hosted applications, but delays of up to 200ms are generally considered acceptable [26].

B. Residual Bandwidth Required to Place New VMs

Our first experiment seeks to illustrate 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 a egress bandwidth of 1Gbps. Given a QoS target of $(0.02s, 0.05)$, with more VMs arriving (recall that all VMs generate SCP traffic) it will be, as illustrated in Fig. 6, 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 $(0.02s, 0.05)$. Given a lower QoS target of $(0.2s, 0.1)$, it is even possible to host up to 5 VMs on the server, as shown in Fig. 6. Placing VMs on the basis of strictly provisioning for VM's 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 level of QoS violation.

In Fig. 7 we depict the change in the level of additional bandwidth to be allocated when a new VM is placed on the server. This is calculated as $\frac{R_{eff}}{|\text{VMs}|}$, i.e. the aggregated effective bandwidth, R_{eff}, divided by the number of allocated VMs, $|VMs|$. Here, we can see clearly the nonlinear nature of effective bandwidth—reflecting the statistical multiplexing of traffic from numerous sources which means that less incremental

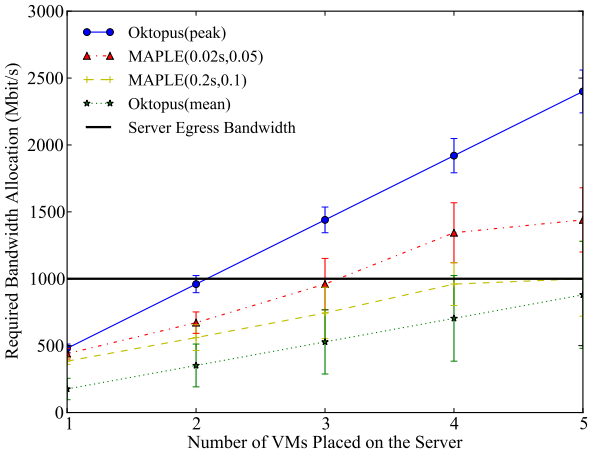


Fig. 6: As additional VMs are placed on a server the required egress bandwidth increases. If peak or mean throughputs per VM are statically allocated as in Oktopus this bandwidth increases linearly. If effective bandwidth estimations are used the rate of increase reduces due to the smoothing effect of the statistical multiplexing of traffic from numerous sources. MAPLE can accommodate more VMs on each server whilst satisfying QoS targets than Oktopus can whilst allocating based on peak throughput.

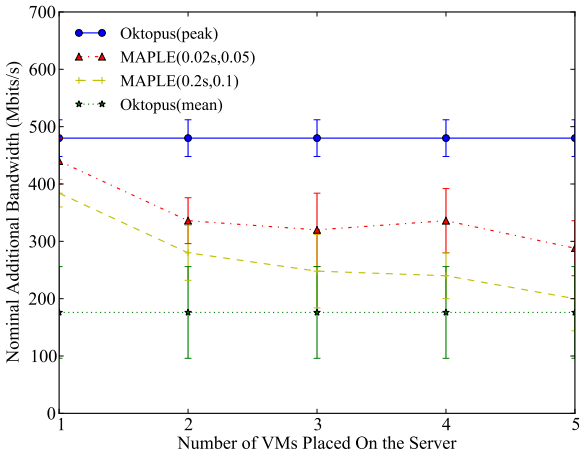


Fig. 7: The additional bandwidth that needs to be allocated when a new VM is placed on a server. MAPLE can place additional VMs based on the availability of less residual bandwidth then can Oktopus using peak throughput estimations since it takes into account the statistical multiplexing effect captured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

bandwidth need be provided to attain the same level of QoS. These results are consistent with the detailed experimental results and theoretical proofs that can be found respectively in Davy *et al.* [20] and Kelly [13].

C. Analysis of QoS Violations

We analyse QoS violations in the emulated datacenter incurred by the VM placement algorithms. Firstly, 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 delay 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 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 none (or tiny) QoS violation levels, while links that face QoS violations suffer with frequent, strong 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 the MAPLE and Oktopus algorithms attempt to allocate VMs in a same subtree with small residual bandwidths in order to saving routing costs, which will frequently result in scenarios where VMs are densely co-located around some links.

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

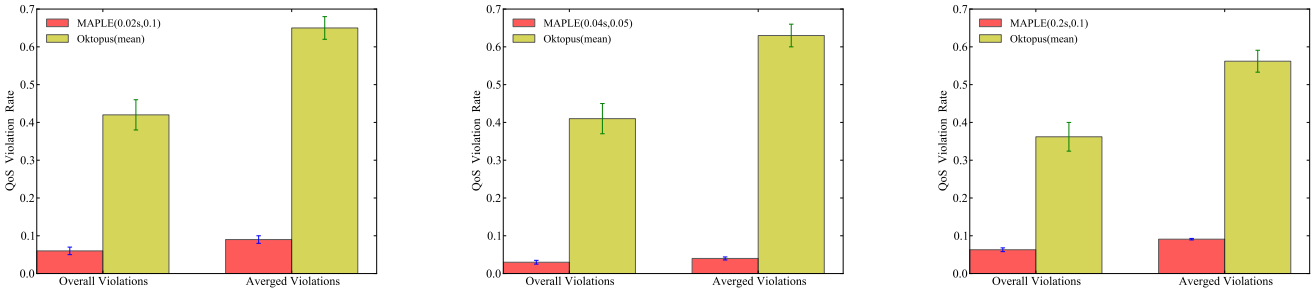
$$\text{averaged\_violation\_rate} = \frac{\text{sum(QoS\_violation\_rates)}}{|\text{links\_with\_violations}|} \quad (3)$$

Fig. 8 depicts the QoS violations where the QoS target is $(0.02s, 0.05)$. 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 the VM placements given by Oktopus allocating using peak throughput result in no QoS violations or tiny violation rates, Fig. 8 only presents results for the other two algorithms. We can clearly see that when allocating using mean throughputs to all VMs, Oktupus tends to have high probability ($> 40\%$) that packets will suffer with packet delay more than 0.02s. In particular, for the links where QoS violations actually occurs, we observe very strong violation rates, on average $> 60\%$. In contrast, MAPLE, which allocates on the basis of effective bandwidth estimates, succeeds in keeping QoS violations within the target range.

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 with three different QoS target $(0.02s, 0.1)$, $(0.04s, 0.05)$, and $(0.2s, 0.1)$, Fig. 9 depicts that MAPLE is always able to meet the target, keeping the violation rates at an acceptable level, whereas Oktopus results in violation rates varying depending on the different delay targets; it never succeeds in keeping the rates within the acceptable range.

D. Analysis of Allocated Bandwidths and VMs

During each of 20 experimental runs, the emulated datacenter was populated with VMs until it was almost fully loaded, i.e., when it could not accept new VM ensemble placement requests. Fig. 10 depicts the total bandwidth allocations made by the three placement algorithms. On average,



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

Fig. 9: QoS violation rates for different delay based QoS targets. MAPLE always succeeds in keeping violations below the specified target level, whereas Oktopus (mean) never does.

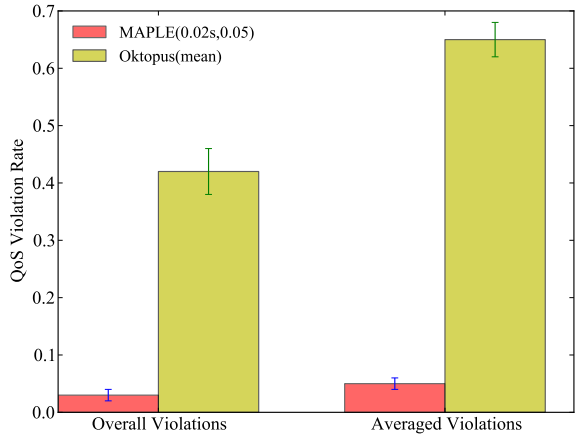


Fig. 8: QoS violation levels for QoS target (0.02s,0.05) for MAPLE and Oktopus. Oktopus over allocates network resources resulting in significant overall and localised levels of QoS violation. MAPLE's used of effective bandwidth estimates means that QoS violations are within the acceptable range.

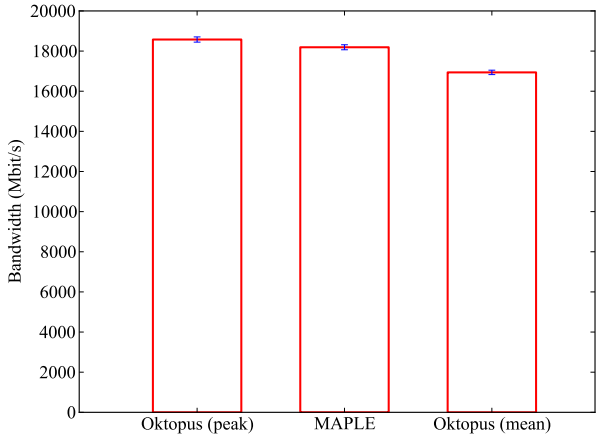


Fig. 10: Total allocated network bandwidth. Oktopus (mean) allocates the least bandwidth, but at the cost of significant QoS violation rates. MAPLE and Oktopus allocate similar levels of overall bandwidth, but MAPLE accommodates significantly more VM placements.

and regardless of the QoS targets, Oktopus allocating based on mean throughputs, produced the most resource-conserving approach, as it managed to placed more VMs (approximately 100) while allocating a smaller amount of bandwidth (sum up to 16,938 Mbps), as shown in Fig. 11 and Fig. 10. In contrast, MAPLE requires more aggregated bandwidths for the placed VMs. Yet, it still managed to place as many VMs as Oktopus using mean throughput places. At a first glance, Oktopus allocating peak throughput allocates aggregated bandwidth of 18,576 Mbps—apparently similar performance of MAPLE. However, from Fig. 11, we see that peak throughput approach placed a significantly smaller number of VMs, compared to the other algorithms. Overall, we conclude that MAPLE is relatively resource-conserving, compared to Oktopus allocating peak throughput, which used more aggregated bandwidths yet placed smaller number of VMs.

VII. Conclusion

We have demonstrated that the Oktopus variant that allocates bandwidths to tenant VMs based on mean throughput tends to suffer with low QoS performance, whereas the 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. Existing network-aware VM placement schemes do not determine this optimal value; they are designed to maximise throughput of datacenter networks, but not to deliver predictable performance in terms of the latency experienced by users of datacenter hosted applications. Our proposed system, MAPLE, provides this form of predictability by placing the 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 an emulated datacenter network suggest that MAPLE succeeds in meeting QoS targets whilst simultaneously allocating both computing and network

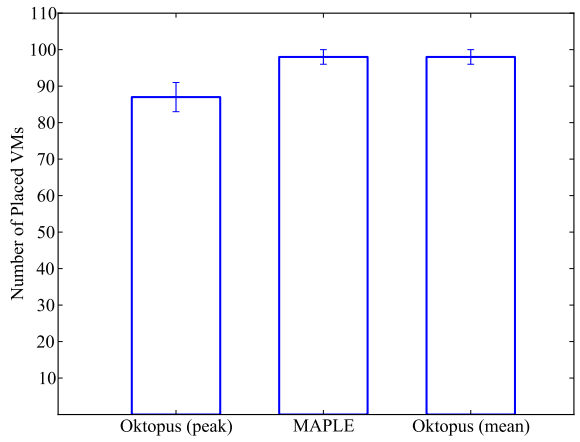


Fig. 11: Total numbers of placed VMs. MAPLE and Oktopus (mean) succeed in placing all VMs. Oktopus (peak) does not as it is limited by the available network bandwidth.

resources in an efficient manner.

Future work will focus on evaluating MAPLE in larger scale deployments with multiple application types with multiple QoS targets. We will explore different approaches for selecting subsets of servers as candidates for placement of incoming VM ensemble requests, with taking into account that anti-colocation is a constraint. Finally, we plan to extend MAPLE to allow it to support elastic VM ensembles in which constituent VMs can be instantiated on-the-fly to handle growing demand.

ACKNOWLEDGEMENTS

We thank Prof. Tilman Wolf (University of Massachusetts Amherst) for providing feedback on early drafts of this paper and Dr. Alan Davy (TSSG, Waterford Institute of Technology) for providing the implementation of the effective bandwidth estimation technique. This work was partially funded by: 1) Science Foundation Ireland via the Research Brazil Ireland project (grant no. 13/ISCA/2843) and via the FAME strategic research cluster (grant no. 08/SRC/I1403); 2) 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 (TGI) project; 3) the Irish Research Council via grant no. ELEVATEPD/2013/26; and 4) by the European Commission via the SOLAS IAPP FP7 project (grant no. 612480).

REFERENCES

[1] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *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.

[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.

[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 IMC*, 2013.

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

[12] Rackspace Cloud. [Online]. Available: http://www.rackspace.com/

[13] F. Kelly, "Notes on effective bandwidths," *Stochastic Networks: Theory and Applications*, vol. 4, pp. 141–168, 1996.

[14] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys Tutorials*, no. 99, pp. 1–20, 2012.

[15] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and System Management*, 2014.

[16] C. Guo, G. Lu, H. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proc. ACM CONEXT*, 2010, pp. 15:1–15:12.

[17] 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.

[18] 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.

[19] 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.

[20] A. Davy, D. Botvich, and B. Jennings, "Revenue optimized iptv admission control using empirical effective bandwidth estimation," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 599–611, 2008.

[21] "VMware vCenter," 2014. [Online]. Available: http://www.vmware.com/products/vcenter-server

[22] N. R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM*, 2009, pp. 39–50.

[23] G. Combs, "Wireshark hompage," http://www.wireshark.org/, 2006.

[24] L. Shi, J. Furlong, and R. Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers," in *IEEE Management of Cloud Systems Workshop (MoCS)*, 2013.

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

[26] Cisco Online Document. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/5125-delay-details.html