Multipath Bandwidth Guarantees for Multi-Tenant Cloud Networking

Wei Wang*†,Yi Sun*, Steve Uhlig‡, Gengfa Fang§, Nanshu Wang*, Zhongcheng Li*
*Institute of Computing Technology, CAS, †University of Chinese Academy of Science
‡University of Queen Marry, UK, §University of Macquarie
E-mail: {wangwei2012,sunyi,wangnanshu,zcli}@ict.ac.cn,Gengfa.fang@uts.edu.au,steve.uhlig@gmail.com

***Abstract*—Resource isolation of the computation and storage in the cloud is relatively mature, but the network resource is still shared among tenants leading to variable and unpredictable network performance when bandwidth guarantees are not enforced. Currently most of the bandwidth guarantee approaches are based on the idea of single-path reservation without fully exploiting the multipath resource, which leads to poor network utilization. In this paper, we propose a multi-path bandwidth guarantee approach called *MultiBand*, which provides bandwidth guarantees by allocating bandwidth across multiple paths. We utilize label-based routing technique to explicitly control the packets' transmission paths, and design a MHTB rate limiter model to split and schedule the traffic over the multiple reserved paths. Besides, Our Multiband solution has the work-conserving property. We evaluated our approach through simulations with realistic topologies and typical traffic patterns. Our results show that MultiBand is able to provide multipath bandwidth guarantees and to achieve higher network utility and tenant throughput compared with those of current approaches.**

***Keywords*—Cloud Data Center Network; Bandwidth Guarantee; Multipath**

I. Introduction

In cloud computing platform, there is a large number of tenants asking for virtual machine (VM) instances including custom OSs, CPU, memory, storage and network *etc.* [1]. These VM instances come with some levels of resource isolation. However, one resource that is not well-isolated in the cloud datacenter is the network resource. Since the shared network can not guarantee the bandwidth for each tenant, applications of the tenants are likely to experience unstable and unpredictable network performance in some cases [18], [22]. This impedes many applications which require guaranteed Quality of Service (QoS) such as the response-time when considering merging into the cloud platform.

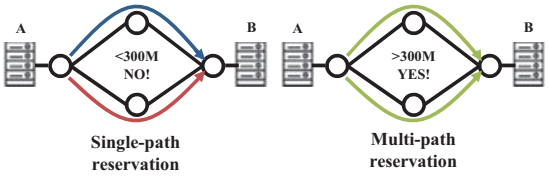
In order to tackle the above problem, some solutions have been proposed recently to provide bandwidth guarantee inside the datacenter. The two main design goals of bandwidth guarantee schemes are: (1) to provide bandwidth guarantee according to user-defined traffic patterns; (2) to dynamically adjust tenant's flow rate to efficiently use the spare bandwidth, *i.e.*, the *work conserving* property [23], [24]. Previous work in datacenter traffic management such as Oktopus [7] and SecondNet [15] apply the static reservation scheme to offer tenants the required bandwidth among the VMs. However, these approaches are insufficient given the dynamics of datacenter traffic [9], [20], and may decrease the network utilization of the datacenter. The latest methods, *i.e.*, EyeQ [6], ElasticSwitch [25] and Hadrian [8] improve the performance of the static reservation schemes by introducing the new work conserving mechanism which improves the network utilization while providing the minimum bandwidth guarantee to its tenants.

Current datacenter network topologies are based on multi-root tree, such as Fat-Tree [4], VL2 and Spine-Leaf [14]. In such topologies, multiple equal-cost paths exist between each pair of VMs. To make full use of network resource, current datacenter schedules traffic over different paths such as ECMP [16], Hedera [5] and MPTCP [26]. Unfortunately, using bandwidth reservation for tenants along a single path [6], [8], [25] results in poor network utilization while the network utilization can be greatly improved by fully exploring the multi-path features of the existing topologies. As a result, neither the tenants nor the datacenter providers are benefiting from and satisfied with the above schemes.

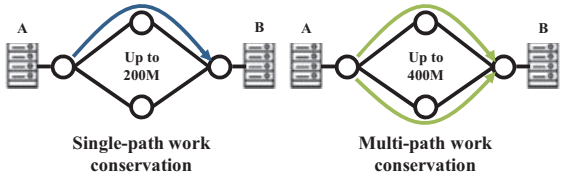
This motivated us to solve the problem of multi-path bandwidth guarantee for datacenter networks. In this paper, we propose *MultiBand*, a multipath bandwidth guarantee solution that provides high network utilization while guaranteeing bandwidth of the tenants. Multiband splits the tenant-required bandwidth and then reserves them over the multiple paths accordingly. The path scheduling and rate control among VMs are done only by the hypervisors based on a multipath hierarchical token-bucket (MHTB) model. We also designed an AIMD-like work conserving approach based on the MHTB model to increase the rate beyond its reserved bandwidth on multiple paths if spare bandwidth (*i.e.*unreserved bandwidth or under-utilized reserved bandwidth) of any used path is available.

We illustrate the benefits of MultiBand compared to current single-path based solutions through a simple example in Fig. 1. Fig. 1(a) shows that multipath bandwidth reservation could fully utilize the network resource and meet more tenants' bandwidth requirements. The tenant-required bandwidths are often various and dynamic [29]. For example, when the traffic increases at peak hours, the tenant requires a higher bandwidth for their VMs. In such situation, Multiband can increase the bandwidth on any available path, while single-path approaches need to find a new path with enough bandwidth if the original one has been fully reserved. Fig. 1(b) illustrates another advantage of MultiBand: better use of available spare bandwidth through the multipath work-conserving property. Traffic in the datacenter is bursty, therefore the links are often underutilized [9]. Since the tenants' traffic flows are transmitted across multiple paths in Multiband, it is able to find larger spare bandwidth to use.

DOI 10.1109/LCN.2016.73



(a) Each link's unreserved bandwidth is 200MB. A tenant needs to apply for 300MB bandwidth from A to B. The required bandwidth can't be reserved on any of the paths by single-path approach, while Multiband can meet this by using two paths.



(b) The bandwidth from A to B is successfully reserved, and currently each link's spare bandwidth is 200MB. Multiband could increase tenant's bandwidth by utilizing 400MB spare bandwidth on two paths.

Fig. 1: An example showing the advantages of multipath-based bandwidth guarantee

In the rest of this paper, we firstly present a brief overview of Multiband, and then discuss the two main features of the Multiband solution in detail, *i.e.*, multipath bandwidth guarantee and multipath work conservation in section II. We then evaluate the performance of Multiband and compare the results with current proposals in section III. Related works will be discussed in section IV followed by conclusions in section V.

II. Multipath Bandwidth Guarantee

Fig. 2 shows the overall architecture of Multiband which will be detailed in this section. In particular, Multiband includes two main components: bandwidth reservation and enforcement, and work conservation aiming at dynamically using the spare bandwidth.

A. Multipath Bandwidth Reservation

Cloud providers usually allow tenants to describe their traffic patterns and specify minimum bandwidth requirements for their VMs. In previous work, different models have been proposed to abstract the tenant's bandwidth requirements. The *hose-model* has been widely used in previous work such as [27], assuming that each VM of a given tenant is connected to a non-blocking switch by a virtual link whose capacity is equal to the VM's minimum required bandwidth. Hadrian [8] proposed a hierarchical hose model to support both intra- and inter-tenant communication patterns. It is well-known that implementing the hose model requires rate enforcement at the granularity of VM-to-VM pairs, while per-VM limiters dont suffice, *e.g.*some of the VM-to-VM flows of VM A are bottlenecked in the hose model at the source side, while other VM-to-VM flows are bottlenecked at the destination side. Thus, it requires VM-to-VM limiters to emulate the hose model. Fortunately, some prior work, *e.g.*, Elasticswitch [25] has proposed methods to partition the hose model, which can be

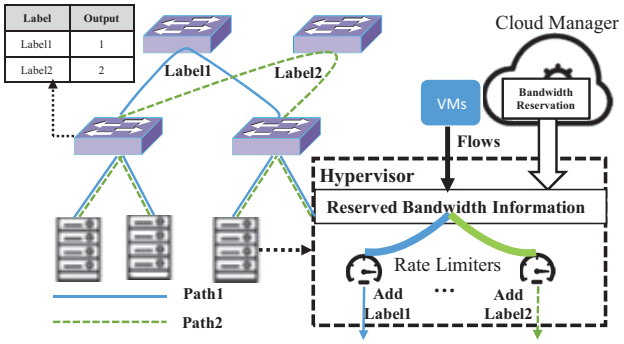


Fig. 2: Multiband Architecture

used in our approach. In this paper, we focus on the problem of how to meet the VM-to-VM bandwidth requirements through multiple paths.

We firstly formulate the multipath bandwidth reservation problem. We assume that VM X communicates with VM Y requiring $B_{X\to Y}$ of network bandwidth. $B_{X\to Y}$ will be guaranteed by splitting bandwidth respectively on $|N|$ paths ($N \subset M$, where M denotes equal-cost path set available between VM X and Y). The allocation of $B_{X\to Y}$ is needed to keep the overall bandwidth reservation of the network load-balanced as much as possible for the following two reasons: (1) to increase overall network utilization while avoiding hot-spot on any path, and (2) to reduce the packet re-ordering incurred by multipath transport (detailed in Section II-B). We measure the load-balanced degree by computing the range (*i.e.*, max-min) of reserved bandwidth value of a path set. Therefore, our objective is to minimize the range of reserved bandwidth in M, when reserving $B_{X\to Y}$ bandwidth on $|N|$ paths selected from M, under constraints that total reserved bandwidth on each link does not exceed its bandwidth capacity. Then we obtain the following formula:

$$\begin{aligned} &minimize\ max(path_m.rsvB) - min(path_m.rsvB) \\ &s.t. \sum_{path_n \in N} path_n.rsvB_{X\to Y} = B_{X\to Y} \\ &l.rsvB < l.capB, \forall l \in path_n \\ &1 \le |N| \le \upsilon \\ &path_m \in M, path_n \in N, N \subset M \end{aligned} \tag{1}$$

where $rsvB$ refers to the reserved bandwidth on a path (We could use (un)reserved bandwidth of a bottleneck link to represent (un)reserved bandwidth of a path). $rsvB_{X\to Y}$ is the reserved bandwidth for communication between VM X and Y, and $capB$ is the total bandwidth capacity of a network link l. υ is an important variable which constrains the higher bound of the number of selected paths (*i.e.*$|N|$). If $\upsilon = 1$, the problem is equal to the previous work that reserves bandwidth on a single path. If $\upsilon > 1$, a proper value $|N| \in [1, \upsilon]$ is selected to assign the bandwidth reservation on N path in set M. Smaller value of $|N|$ is preferred, since despite larger value of $|N|$ leverages more available paths, it does come with obvious drawbacks: frequent packet re-ordering and increasing complexity on the traffic management. We show

in Section III that a relatively small value such as 2 or 4 is sufficient to obtain expected benefits.

Before describing the multipath bandwidth reservation algorithm, we firstly define an important data structure, *struct Bandwidth_Require* detailed as below which is associated with the *hose-model*, VM allocation algorithm, bandwidth allocation algorithm and the enforcement component in hypervisors.

```
struct Bandwidth_Require
{
    int srcVM; //src VM id
    int dstVM; //dst VM id
    float bandwidth; //total required bw

    int srcHost; //src VM host id
    int dstHost; //dst VM host id

    int pathLable[N];//reserved paths labels
    int pathBw[N]; // bw on each path
};
```

Algorithm 1 Multipath Bandwidth Reservation Main Loop

```
1: while True do
2:     Struct Bandwidth_Require BRVector
3:     Translate tenant's hose-model and fill BRVector
4:     for req in BRVector do
5:         req.SrcHost, req.DstHost =VMAlloc(req)
6:         req.pathLable, req.pathBw =BwAlloc(req)
7:         UpdateNetwork(req)
8:     end for
9:     Distribute BRVector to corresponding hosts
10: end while
```

Algorithm 1 tells how multipath bandwidth reservation works. When processing tenant requirements of bandwidth reservation, *src VM*, *dst VM*,*bandwidth* is firstly translated from *hose-model*. For each requirement, function `VMAlloc` then decides the two VM's hosts (line 5). Note that the specific implementation of `VMAlloc` may apply different policies once it is sticking to the rule: there should be enough unreserved bandwidth between the two hosts. Then function `BwAlloc` (detailed in Algorithm 2) will return the allocated N paths and the values of corresponding bandwidth on each path (line 6). The cloud orchestration (controller) maintains a global bandwidth reservation information of each link, which is updated by function `UpdateNetwork` (line 7). At last, After filling all information in the *BRVector*, it is sent to the corresponding host's hypervisor which performs the bandwidth enforcement.

Algorithm 2 shows the details about the multipath bandwidth allocation algorithm. The input of the function `BwAlloc` is the bandwidth requirement information *req*, and the outputs include the paths to be reserved and how much bandwidth reserved on each path. At first, the available paths M between the src-host and dst-host is obtained by the function `GetPaths` (line 2). Then M is sorted according to the value of reserved bandwidth of each path in an increasing

Algorithm 2 Multipath Bandwidth Reservation Algorithm

```
1: function BWALLOC(struct Bandwidth_Require req)
2:     M =GetPaths(req.srcHost, req.desHost)
3:     LeftB = req.bandwidth
4:     Sort(M)
5:     for i from 1 to v do
6:         req.pathLable[i − 1] = M[i − 1].pathLabel
7:         next = (i == v)?M[i − 1].capB : M[i].rsvB
8:         if LeftB > i ∗ (next − M[0].rsvB) then
9:             AddB = next − M[0].rsvB
10:            Flag = 0
11:        else
12:            AddB = LeftB/i
13:            Flag = 1
14:        end if
15:        for j from 0 to i − 1 do
16:            if M[j].rsvB + AddB > M[j].capB then
17:                return null, null        ▷ capacity is used up
18:            end if
19:            M[j].rsvB = M[j].rsvB + AddB
20:            req.pathBw[j] = req.pathBw[j] + AddB
21:        end for
22:        LeftB = LeftB − i ∗ AddB
23:        if i == v && LeftB! = 0 then
24:            return null, null            ▷ capacity is used up
25:        end if
26:        if Flag == 1 then
27:            return req.pathLable, req.pathBw
28:        end if
29:    end for
30: end function
```

order (line 4). Lines from 5-25 allocate the required bandwidth to the paths to achieve the most load-balanced reservation. In particular, it greedily increases the number of paths, i, from 1 to the upper bound v. In each round it checks whether the unreserved bandwidth of first i paths in M is enough to allocate the required bandwidth value with the constraint that after reservation each path's reserved bandwidth of the first i paths won't exceed the $(i+1)th$ path's reserved bandwidth (line 6-14). Once this is satisfied, it returns the first i paths and the allocated bandwidth for each one; otherwise, it records currently reserved bandwidth on each path (line 15-25) and increases the path number i. Since the algorithm tried all the available paths and in each round it selects the least reserved paths to allocate the bandwidth so that the output could guarantee the load-balanced reservation for relatively small number of paths. The time complexity is $O(M \log^M + v^2)$, and generally, $|M|$ and v are both small values, *e.g.*in a 16-ary Fattree, the number of $|M|$ is only 64 and a small value of v (2 or 4) could achieve the needed performance improvement as showed in our experiments.

B. Enforcement

To accurately enforce the reserved bandwidth over the network, Multiband needs to explicitly control the packets' transmission paths. Most related work of bandwidth enforcement relies on the rate limiters not only on the host hypervisors

but also on the switches [6], [24]. Therefore switches have to maintain large amount of bandwidth reservation information of VM pairs. Once new bandwidth requirements are coming (which are very frequent in the Cloud), the corresponding switches need to be re-configured. Since we split the bandwidth across multiple paths, bandwidth enforcement relying on switches will introduce heavy workload on the already busy switches.

In this paper, we apply an explicit path control approach which is based on the *label*-based routing technique and deploy rate limiters only on hypervisor side. In current cloud datacenter topologies, the path information between any two hosts can be pre-configured, therefore each path could be assigned with a *lable*, which is used to route the packets to the destination along the associated path, such as SecondNet, VL2 [14].

There are different ways to encode the routing information in the label. One is to encapsulate the original packets in the MPLS header and use the stack of MPLS labels to carry a list of output port index for each hop [15]. So the switches only need to forward the packets to the port that is specified in the header. The advantage of this approach is that the switch doesn't need to store too much forwarding information, however it requires switches to support the port-switching [15]. Another way is based on the IP-in-IP tunnel and can be deployed in commodity switches as proposed in [17]. The label is designed as the tunnel IP, the switches will use LPM (Longest Prefix Matching) to route the packets. Since the number of available paths is extremely large in large scale networks, and the number of path labels will significantly exceed the size of table entries in switches. However we can use compression algorithms to reduce the label numbers. For example, as presented in [17] for Fattree(64) the paths can be installed in switches with 64K table entries.

The routing information is calculated centrally by the Cloud controller which is pre-installed in the switches. When the Cloud controller creates new VM instances in the hypervisor, the reserved paths' *labels* and corresponding bandwidths of each path are also sent to the hypervisor, as shown in Algorithm 1 line 9. The hypervisors use this information to control the paths and limit flow rate on each path.

Despite that many hypervisor-based rate limiters have been proposed [25], [28], in our MultiBand, for every VM pair, we have multiple rate limiters with each corresponding to a reserved path. In order to support multipath rate limiting, we extend hierarchical token-bucket (HTB) (such as Linux TC HTB qdiscs, [3]) to multipath HTB (MHTB) model, illustrated in Fig. 3. Each VM pair corresponds to a MHTB model which has two levels of rate limiters: root limiter and leaf limiter, see Fig. 3. A root rate limiter is to guarantee and specify one VM's aggregate bandwidth limitation. The aggregate traffic is then splitted by the *classifier*, and forwarded to the paths' rate limiters, *i.e.*the leaf limiters. The leaf limiters use the token bucket for flow control and its rate is set to the value of the reserved bandwidth on corresponding path. The path label will be filled in the packet header at the leaf limiter as described previously.

The root limiter has a *classifier* rule table which contains the rules to assign the split traffic to each leaf limiter. The rule structure is $\langle$*source-port, leaf-id, last-sent-time, expire-time*$\rangle$. Since the source and destination IP addresses in a MHTB model remain the same, only *source-port* number is needed to identify a flow. The *source-port* is the matching field, and the matched packets are forwarded to the leaf *leaf-id*.

The rules of *classifier* are mainly designed (1) to schedule flows to less congested available paths and (2) to reduce the packet reordering as much as possible. Initially, the rule table is empty. If a packet arrives at the root limiter without matching any rule, a new rule is added and its *leaf-id* is set to the limiter of least-congested path [1]. *last-sent-time* is the time when the last packet matching this rule was sent and *expire-time* stores the max latency difference among the paths. If *last-sent-time* + *expire-time* < *current-time*, this rule could be updated with new *leaf-id* that is less-congested. Specifically, if the time between a new coming packet and the last-sent packet of the same flow is longer than maximum latency difference, the packet can be forwarded to any other paths without worrying about packet reordering. Similar ways are described in FLARE [19], where it is proved that packet reordering caused by multipath delivery can be reduced significantly.

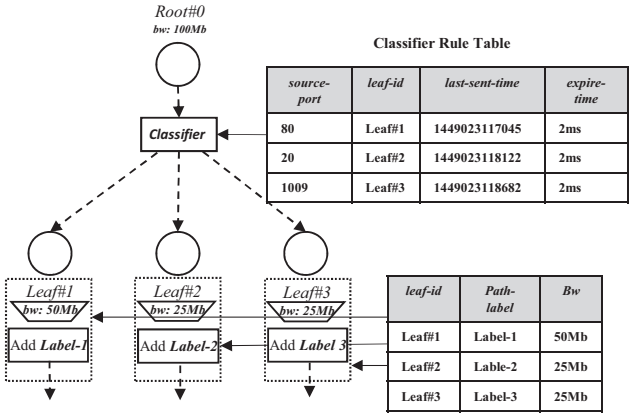


Fig. 3: Multipath rate limiter

In most cases, if a flow's rate can be met in a single path, only one element will be in *leaf-id* of the rule. However, if the rate of a large flow can not be met on a single path, there will be multiple leaf limiters filled in the *leaf-id* field. In such cases, the packets of a flow are sent on these paths simultaneously, which may cause packet reordering.

However in the datacenter network, the effect of such packet re-ordering is very limited. In particular, there are two reasons why reordered packets can lead to TCP's suboptimal performance: (1) Head of Line(HOL) blocking in TCP, *i.e.*, if a sequence of packets are arrived except the first one, all the following packets will be blocked in the buffer and can't be passed to application, which will also trigger congestion control in the sender; (2) Spurious retransmission, that since TCP can not distinguish reordered packets from lost packets,

[1]We use the latency of a path(obtained by periodically ping) to show the congestion degree

it will perform fast retransmit once 3 duplicate ACKs arrive for the same packet. Thus, the out-of-order delivery will cause spurious retransmission resulting in performance degradation. However, the following observations indicate that Multiband in datacenter can avoid the above two inducing factors, and is unlikely to result in significant packet re-ordering and consequently won't affect TCP's performance.

Observation 1: The number of hops between hosts in datacenter is relatively small (typically 6-8), and the traffic is load-balanced among the paths, so the difference of latency between paths is small. Therefore the HOL blocking time is very short even when a few packets are out of order [13].

Observation 2: 3-duplicated-ACKs-based fast retransmissions are gradually removed from TCP in datacenter network due to its spurious retransmission problem. Instead, TCP could handle small scale and short-time re-ordering by dynamic DupACKtreshold, DSACK or timestamp options of TCP [10].

C. Multipath Work Conservation

Work conserving aims to make full use of the spare bandwidth by dynamically increasing the VM's sending rate over its reserved bandwidth if this is the bottleneck while not bringing any negative impact to the traffic of other VMs. Our proposed Multiband solution comes work conserving ability to support multiple paths rather than single path. To prevent any side effect from the rate increasing, we need to collect information about spare bandwidth in terms of congestion degree.

In order to reduce workload on switches, an end-to-end approach to detect the spare bandwidth is applied in Multiband. In particular, ping packets are sent periodically on the multiple paths between a VM pair and then the max and min RTT (*i.e.*max_rtt, min_rtt) for each path are recorded. We use $C = \frac{rtt - min\_rtt}{max\_rtt - min\_rtt}$ to represent the congestion degree of a path. If $C < threshold$, we can predict that there are spare bandwidths on that path, and the rate can be increased on top of the reserved one. Thus, we can add a new excess bucket (called *EB*) for the corresponding leaf rate limiter, as showed in Fig. 4. The token in *EB* is used only when the token in original bucket (*OB*) is running out. The rate of *EB* is initialized with a very small value, and updated according to the way similar to the TCP's AIMD. The rate is increased by δ Mbps upon positive feedback (*e.g.*, no packet loss for a while). If negative feedback is received (*e.g.*, packet loss or $rtt \approx max\_rtt$), the rate is decreased by a multiplicative factor θ. Once there are 3 packet losses on the path, the rate of *EB* is set to 0. Each path between the VM pair has an excess bucket operating independently in Multiband for work conserving. Since the VMs are more likely to find spare bandwidths on multiple path, the dynamic network utilization is obviously improved compared to that of the single path reservation scheme, as showed in our experiments.

Using bandwidth more than the reserved has to be done with caution because if a link is fully reserved, a small rate increase of any VM through this link will affect the reserved bandwidth of other VMs. To solve this problem, we classify packets into two classes. The first class is normal packets, *i.e.*, the packets which get the token from the original bucket. The second class is the aggressive packets, *i.e.*, the packets which get the token from the excess bucket when the original bucket is used up. These two classes of traffic can be distinguished by switches if the ToS field in the IP header is set with different priorities. Normal packet is granted higher priority with lower delay and higher throughput while the aggressive one has lower priority. Most industrial switches support priority-based multiple queuing, and if the aggressive packets and normal packets compete for the limit queue space, the aggressive packets are dropped with higher possibility. In this way, we can ensure that when trying to use spare bandwidth, the normal traffic of other VMs will not be negatively affected.

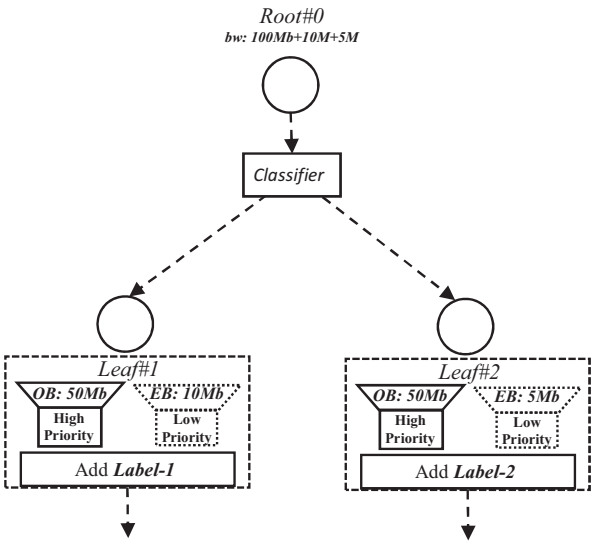


Fig. 4: Work conservation in MHTB model

III. Evaluation

In this section, we evaluate the performance of the next two properties of the MultiBand through experiments: (1)Validity: MultiBand is able to provide bandwidth guarantees and enforcement through multiple paths; (2) Efficiency: MultiBand can provide bandwidth guarantees for more tenants' requests and achieve better performance in terms of both network utilization and VM throughput which are the interest of both cloud providers and customers.

Experiment setup: we use CloudSim, a well-known open source simulator which provides a generalised and flexible simulation framework for Cloud computing infrastructures and applications [2]. We adapted the *Switch* module in CloudSim to support label-based routing. The underlying network topology is a Fat-tree [4] with 4 core switches, 4 pods and each edge switch connects to 2 physical servers[2] as illustrated in Fig. 5. The topology has been configured with multiple oversubscription ratios. Each VM runs an application to generate TCP traffic with varying flow rates and two different traffic patterns. Our TCP version is the TCP Cubic with DSACK enabled.

[2]We believe the conclusions should hold the same result for larger-scale or other types of topologies

Traffic Pattern 1: The sender of each VM pair generates packets at a fixed rate larger than the reserved bandwidth leading to the situation where traffic fully occupies links capacity.

Traffic Pattern 2: A more common situation is that the sender generates packets at a varying rate which follows the unpredictable and burst traffic pattern in [9], thus link is unsaturated for most of the time which will trigger work conserving property.

We compare Multiband with the single-path-based bandwidth guarantee approach (referred as *Singleband*), and pure multipath load balancing (referred as *Load Balancing*). Singleband reserves the bandwidth on single path, and also uses label-based routing to enforce the allocated bandwidth in the network. Singleband will also search for available spare bandwidth on the reserved path when its reserved bandwidth is fully utilized. Our implementation of Singleband can be seen as an idealized version of current solutions such as Seawall, Oktopus and Gatekeeper. Load Balancing schedules the traffic across multiple paths to fully exploit the network resource, the implementation of which is similar to Hedera [5] to schedule long flows from congested paths to idle paths.

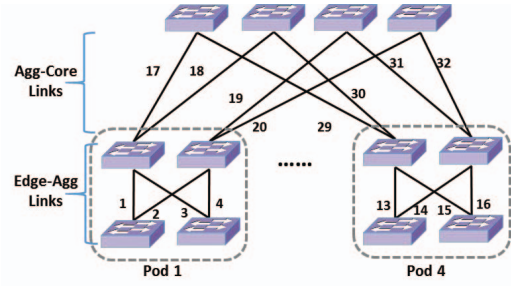


Fig. 5: Network topology

Parameters: In our experiments, the number of available paths between each inter-pod VM pair is 4 ($|M| = 4$). The upper bound υ of $|N|$ is set to 2 or 4. The $threshold$ for spare bandwidth detection is set to be 0.6. For the rate adjustment in work conservation, the bandwidth increment constant δ is 0.5 Mbps, and the rate decrease factor θ is 0.5.

A. Validity of Multiband

In this experiment, we evaluate the validity of Multiband from the perspective of both VMs and the network. In particular, the topology is showed as Fig. 5, and every switch is typically connected with 10 physical machines. During the experiments, we continuously adds bandwidth requests into the network. It is supposed that the VMs of any one pair will not be allocated in the servers of the same pod by the VM allocation algorithm. For simplicity, we randomly select 16 VM pairs to illustrate the results. Table I shows the bandwidth requirements of the selected VM pairs. Path number $|N|$ is fixed to 2 in order to see clearly how bandwidth allocation works in Multiband. Algorithm 2 successfully allocates the required bandwidths of all VMs pairs through 2 paths with the details in Table I. We verify the performance of Multiband in high competing situation with *Traffic Pattern 1*.

Fig. 6 shows throughput measured in experiment for all the VM pairs and links including the ones connecting edge and aggregate switches(edge-aggregate), aggregate and core switches(aggregate-core). Without implementing any limiters on switches, bandwidth enforcement exclusively on hypervisors guarantees bandwidth between VM pairs and limits throughput using the allocated bandwidth of each link as we expected. For instances, Algorithm 2 allocates 2 paths for 500Mbps reservation requests of VM pair 1: VP1-1 `link1-link17-link21-link7` (referring to Fig. 5) with 330Mbps and VP1-2 `link1-link18-link22-link7` with 170Mbps. Fig. 6(a) shows that throughput of VM pair 1 is consistent with reserved bandwidth on each allocated path. In Fig. 6(b) and Fig. 6(c), both links of VP1-1(colored in green) are occupied at 330Mbps and links of VP1-2(colored in red) are occupied at 170Mbps, tantamount to reserved bandwidth for corresponding path. While Fig. 6 provides only one example and is no proof of itself, we tried many different experiments (*e.g.*, different bandwidth requirements of VM pairs, which we do not report due to space limitations), and they all confirm that Multiband is capable of guaranteeing each VM's requested bandwidth.

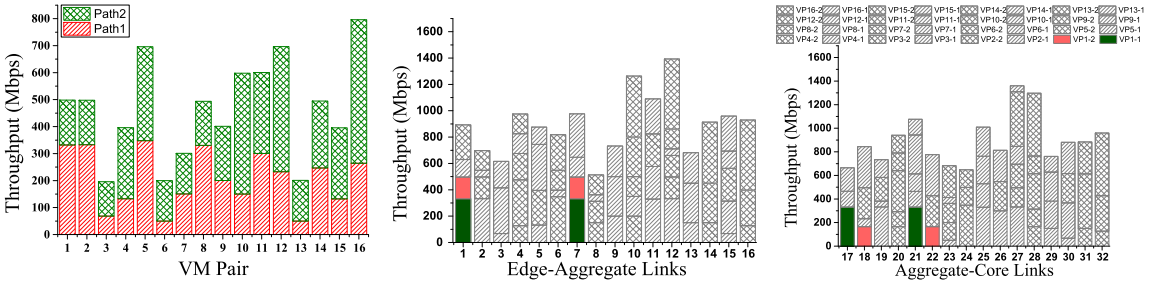
That achieved throughput of both VMs and links strictly matching the reserved bandwidth is assured for two reasons: (1) The MHTB model correctly splits the traffic on each path according to the reserved bandwidth, and enforces the packets' transmission path using label-based routing; (2) The side-effect of packet reordering is very limited. Another experiment is conducted to prove this, in which we count the duplicated ACK (caused by re-ordering packets), which shows only 2.1% of total data byte are acked more than 3 times (fast retransmission threshold). In other words, the penalty brought by packet reordering is negligible compared to the better usage of the total aggregate available bandwidth across multiple paths.

B. Efficiency of Multiband

We now compare the efficiency of Multiband with the Singleband and Load Balancing. For Multiband, in order to evaluate the effect of path number N, we set the value of υ, *i.e.*, the upper bound of path number $|N|$, to 2 (called as Multiband-2) and 4 (called as Multiband-4). In current cloud datacenters, the network links are usually over-subscribed. Therefore, we conduct our experiments on the fat-tree topology with different and typical over-subscription ratios, *i.e.*, 1:1, 1:2 and 1:4 [8]. We continuously add new VM pair requests with random bandwidth requirement in the first experiment. Flow rates of VM pairs follow *Traffic Pattern 1* to simulate the most intensified scenarios. We constantly monitor whether the required bandwidth is guaranteed. We conservatively believe the bandwidth guarantee is failed if throughput is under required bandwidth for more than 1 second. Fig. 7 shows the relationship between the number of guaranteed requests (y-axis) and different over-subscription ratios (x-axis) where Multiband achieves a larger amount of guaranteed tenant requests compared with Singleband and Load Balancing for all over-subscription ratios. Noticeably, MultiBand increases

TABLE I: Reserved bandwidth of 16 randomly selected pairs of VM (Mbps)

Pair No.	1	2	3	4	5	6	7	8
$TotalBw(Bw_{path1}, Bw_{path2})$	500(330,170)	500(330,170)	200(65,135)	400(130,270)	700(350,350)	200(50,150)	300(150,150)	500(300,200)
Pair No.	9	10	11	12	13	14	15	16
$TotalBw(Bw_{path1}, Bw_{path2})$	400(200,200)	600(150,450)	600(300,300)	700(230,470)	200(50,150)	500(250,250)	400(120,280)	800(270,530)



(a) VM-VM throughput (b) Throughput on each edge-agg link (c) Throughput on each agg-core link

Fig. 6: The validity of bandwidth guarantee of Multiband

up to 67% of guaranteed requests for ratio 1:4 compared with Singleband. In addition, higher the over-subscription is, larger the relative difference between Multiband and other approaches is, as the network is prone to suffer congestion. Load Balancing is sensible to congestions due to its lack of bandwidth guarantee and protection of traffic, which leads to very poor performance in terms of guaranteed requests in over-subscribed network. We also observe that the difference between Multiband-2 and Multiband-4 is marginal. Indeed, the main gain of multipath comes from being able to use more than a single path, then the law of diminishing returns kicks in. Therefore, Multiband doesn't need a very large N to provide most of its benefits.

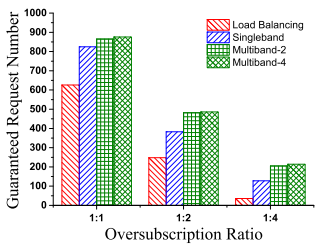


Fig. 7: Number of guaranteed requests for different over-subscription ratios

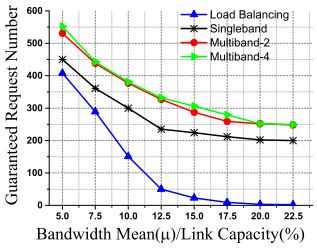


Fig. 8: Number of guaranteed requests for different bandwidth expectation

We then investigate the performance in terms of the bandwidth guarantee when required bandwidth load increases. We randomly generate the value of tenant's bandwidth requests according to Gaussian distribution $(N(\mu, 1))$ with different mean (μ) (the value less than 1Mbps is discarded). The value of the mean μ varies between 5% and 22.5% of the link capacity. The network over-subscription ratio is set to 1:2. As showed in Fig. 8, the smaller the value of μ, the higher the number of guaranteed requests is. When μ increases, Load Balancing scheme struggles to fulfill the tenant's requests and fails to guarantee any request when bandwidth mean is approximately 17.5% of the link capacity. As already mentioned, this is because Load Balancing scheme does not provide guarantee. Bottleneck links will be occupied by the more aggressive VMs, and negatively affect the other traffic (more detailed results for this are showed later). MultiBand outperforms Singleband with more than 25.3% of guaranteed requests thanks to its ability to make better use of the total available bandwidth in congested network.

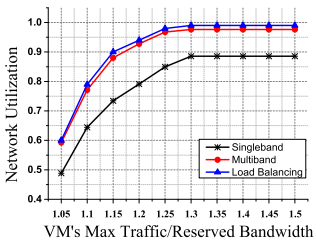


Fig. 9: Network utilization

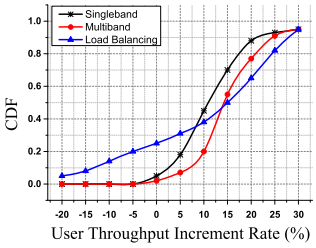


Fig. 10: CDF of user throughput increment ratio

We next study the performance of work conserving of the three schemes selected, especially network utilization which is of particular interest to cloud providers. We first evaluate the network utilization under different traffic loads. *Traffic Pattern 2* is adopted in work conserving experiment to simulate the scenario where spare bandwidth is occasionally available. Besides, we control the aggressiveness of the sender VM by different ratios between the VM's max traffic rate and the reserved bandwidth (x-axis in Fig. 9). Network utilization of all three considered schemes increases when traffic load is increased. MultiBand achieves at least 10.1% higher network utilization than that of Singleband. This stems from the limited options that Singleband considers with only one path, decreasing its ability to be work-conserving. Load Balancing performs the same as MultiBand in terms of network utilization, but as we have already seen at the cost of rejecting lots of tenant requests.

Fig. 10 showing the CDF of the *VM throughput increment*, defined as $(avgThr - requestedBW) * 100/requestedBW$, exposes the fatal weakness of Load Balancing. About 25% of VMs whose throughput is lower than their requested bandwidth for Load Balancing, while still about 19% of the VMs receive a throughput about 25% higher than their requested bandwidth. This means that Load Balancing can't provide bandwidth guarantee for a significant fraction of the VMs, and the aggressive VMs will obtain more bandwidth than they need while penalizing other VM's.

Fig. 10 also illustrates that, as expected, through the work conserving property, Multiband as well as Singleband not only guarantees bandwidth requests, but also improves the throughput of nearly all VM pairs. MultiBand achieves higher VM throughput increment compared to Singleband: With Multiband, 80% of VMs get more than 10% of throughput compared to 55% of the VMs for Singleband. Leveraging multiple paths allows MultiBand to better exploit the network bandwidth resources.

IV. Related Work

Bandwidth reservation and guarantees have been studied for several decades. The seminal DiffServ [12] and IntServ [11] models have been proposed to provide service differentiation and bandwidth guarantees respectively. However, both DiffServ and IntServ need to either set parameters or to maintain states on routers. In cloud datacenter networks, the situation is different, as the topology is known and under control by the operator allowing for much simpler approaches. We now review the related work that focuses on the bandwidth guarantees for tenants in cloud datacenter networks.

Oktopus is proposed to provide predictable datacenter network performance in [7]. Oktopus assumes the hose model for tenants requests and relies on a centralized approach to provides bandwidth reservations and enforcement for each tenant. However, it is not work-conserving and is not desirable for cloud providers given the bursty nature of datacenter traffic. This leads to poor utilization of the network. Similar to Oktopus, SecondNet proposed a solution to guarantee the bandwidth for each VM pair in [15]. SecondNet also relies on a centralized controller to control each VM's path and rate which significantly limit the scalability of the approach. Furthermore, SecondNet is not work conserving as well.

EyeQ [6] provides a solution with both bandwidth guarantees and work-conserving. EyeQ assumes that the core network of the datacenter is congestion-free. This assumption allows EyeQ to deploy the rate limiters only in the hypervisors without the need to enforce the rates inside the network. However, it is known that real datacenters are not congestion-free and the traffic is bursty [9]. GateKeeper [27] also assumes a congestion-free core network.

ElasticSwitch [25] and Hadrian [8] are the latest work in the work-conserving category. They both rely on a hierarchical hose model to provide inter-tenant minimum bandwidth guarantees. However, Hadrian needs dedicated switches to count the frequently varying flow numbers of each tenant, which might be impractical in current datacenter.

CloudMirror [21] proposes to rely on Tenant Application Graphs (TAG) to leverage the tenants knowledge of their traffic pattern instead of the hose model. Generating these TAGs is however a big challenge for cloud operators as some tenants might not even know themselves traffic pattern in practice.

V. Conclusion

Bandwidth guarantees and high network utilization are the main concerns from the perspectives of cloud tenants and providers respectively. Existing datacenter network traffic management schemes rely on single-path based resource reservation, which leads to poor network utilization. Considering the rich multipath resources in current datacenter networks, we proposed Multiband, a multipath-based work-conserving bandwidth guarantee solution. Multiband reserves bandwidths on multiple paths without causing significant packet reordering. The bandwidth enforcement only runs in the hypervisor and requires no rate limiters on the switches. Multiband is also work-conserving, and therefore achieves high flow rates and network utilization by using the spare bandwidth across multiple paths. Our simulations show that Multiband achieves higher network utilization and tenants' throughput than the current single-path based solution and the classical multipath load-balancing scheme.

VI. Acknowledgement

The project was sponsored in part by grants from National Basic Research Program of China (973) under Grant No. 2012CB315802, National Natural Science Foundation of China under Grant No. 61379133, joint project with RICOH, Natural Science Foundation of Inner Mongolia under Grant No.2015MS0601, and National Project of Culture and Technology Promotion Plan of China under Grant No.201201-02.

References

[1] Amazon aws products. In *http://aws.amazon.com/products/*.

[2] Cloudsim. In *http://www.cloudbus.org/cloudsim/*.

[3] Linux tc. In *http://linux.die.net/man/8/tc*.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. volume 38, pages 63–74, 2008.

[5] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI 2010*, volume 10, pages 19–19, 2010.

[6] M. Alizadeh, C. Kim, A. Greenberg, V. Jeyakumar, D. Mazires, and B. Prabhakar. Eyeq: practical network performance isolation at the edge. 2013.

[7] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. *ACM SIGCOMM*, 41(4):242–253, 2011.

[8] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. O'Shea. Chatty tenants and the cloud network sharing problem. *Proceedings of Usenix Nsdi Lombard IL*, 2013.

[9] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *IMC 2010*, pages 267–280, 2010.

[10] E. Blanton, M. Allman, and E. Blanton. Using tcp duplicate selective acknowledgement (dsacks) and stream control transmission protocol (sctp) duplicate transmission sequence numbers (tsns) to detect spurious. *IETF RFC 3708*, 2004.

[11] R. Braden. Resource reservation protocol (rsvp) – version 1 functional specification. *Ietf Rfc*, 40(5):116 – 127, 1997.

[12] M. Carlson. An architecture for differentiated services. *Ietf Rfc*, 1998.

[13] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella. On the impact of packet spraying in data center networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 2130–2138. IEEE, 2013.

[14] A. Greenberg, J. R. Hamilton, N. Jain, K. Srikanth, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: A scalable and flexible data center network. In *In SIGCOMM*, 2009.

[15] C. Guo, G. Lu, H. J. Wang, and etc. Secondnet: a data center network virtualization architecture with bandwidth guarantees. *Acm Conext*, 24:620–622, 2010.

[16] C. E. Hopps. Analysis of an equal-cost multi-path algorithm. 2000.

[17] S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao, and C. Guo. Explicit path control in commodity data centers: Design and applications. In *NSDI 15*, pages 15–28, Oakland, CA, May 2015. USENIX Association.

[18] K. Jang, J. Sherry, H. Ballani, and T. Moncaster. Silo: Predictable message latency in the cloud. *Acm Sigcomm Computer Communication Review*, 45(5):435–448, 2015.

[19] S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic load balancing without packet reordering. *Acm Sigcomm Computer Communication Review*, 37(2):51–62, 2007.

[20] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The nature of data center traffic: measurements & analysis. In *IMC 2009*, pages 202–208, 2009.

[21] J. Lee. Application-driven bandwidth guarantees in datacenters. 2014.

[22] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. *Proceedings of Annual Conference on Internet Measurement*, 15(2):50 – 53, 2010.

[23] J. C. Mogul and L. Popa. What we talk about when we talk about cloud network performance. *Acm Sigcomm Computer Communication Review*, 42(5):44–48, 2012.

[24] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. Faircloud: sharing the network in cloud computing. In *Proceedings of the ACM SIGCOMM 2012*, pages 187–198, 2012.

[25] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos. Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing. volume 43, pages 351–362, 2013.

[26] C. Raiciu and S. e. a. Barre. Improving datacenter performance and robustness with multipath tcp. *ACM SIGCOMM*, 41(4):266–277, 2011.

[27] H. Rodrigues. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. *Usenix Wiov*, 2011.

[28] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. In *In NSDI*, 2011.

[29] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. The only constant is change: incorporating time-varying network reservations in data centers. *ACM SIGCOMM*, 42(4):199–210, 2012.