IEICE TRANS. INF. & SYST., VOL.E99–D, NO.3 MARCH 2016

LETTER

# Dynamic Inbound Rate Adjustment Scheme for Virtualized Cloud Data Centers

**Jaehyun HWANG**[†], **Cheol-Ho HONG**[††], *Nonmembers, and* **Hyo-Joong SUH**[†††a)], *Member*

**SUMMARY** This paper proposes a rate adjustment scheme for inbound data traffic on a virtualized host. Most prior studies on network virtualization have only focused on outbound traffic, yet many cloud applications rely on inbound traffic performance. The proposed scheme adjusts the inbound rates of virtual network interfaces dynamically in proportion to the bandwidth demands of the virtual machines.

***key words:*** *network virtualization, dynamic rate adjustment, inbound data traffic, cloud application*

## 1. Introduction

Virtualization has emerged as a key technology in various areas, e.g., cloud data centers, since it can consolidate multiple virtual machines (VMs) into one physical machine. In this area, the main research issues have been CPU and VM scheduling [1] when running multiple VMs. Recently, many researchers have focused on network I/O scheduling where a different amount of bandwidth is allocated to each VM [2], [3]. Network bandwidth itself is a very important resource to provide differentiated quality of service (QoS) in cloud environments. However, most of the prior studies have only focused on static outbound traffic in designing such bandwidth allocation or reservation schemes [4]–[6], whereas the overall service quality is determined by the dynamic inbound traffic performance of many cloud applications, e.g., Hadoop and web search. In these cloud applications, an aggregator node generally gathers response data from many worker nodes simultaneously following the partition/aggregate design pattern [7]–[10].

In this paper, we propose a dynamic rate adjustment scheme among multiple virtual network interface cards (vNICs) for inbound data traffic. To the best of our knowledge, this is the first attempt that considers a differentiated rate adjustment for dynamic inbound traffic in network device virtualization. Basically, it is difficult for the local hypervisor to control the amount of traffic coming from the outside world, i.e., the amount of inbound traffic, without any cooperation from outside clients. For this reason, we leverage a TCP flow control scheme, an operating system (OS)-level technique, which has been proposed for data center networks [8]–[10]. Such a scheme works on a guest OS of the local host, and it controls the sending window size of remote peers so that the total amount of incoming traffic does not exceed the physical link capacity of the local host. However, this scheme works under a precise link capacity, while the vNIC capacity varies as a result of competition for resources with other vNICs. This necessitates explicit rate adjustment among vNICs to transparently reveal the appropriate link capacity to the guest OS. One of the easiest ways to achieve this goal is to simply assign a static capacity to each vNIC. Unfortunately, this static assignment may result in under- or over-utilization due to varying bandwidth demands of VMs. Therefore, the rate of the vNICs needs to be adjusted dynamically to achieve efficient network resource utilization.

In short, when multiple VMs share a physical NIC (pNIC) through their vNICs, we face two problems:

- How can we assign/adjust the rate of each vNIC for inbound traffic?
- When is the right time to perform a rate adjustment?

In the following sections, we describe static rate assignment and our proposed dynamic rate adjustment method, answering the above questions.

## 2. Static Rate Assignment

Let $C_{pNIC}$ and $C_{vNIC_i}$ denote the capacity of the pNIC and the $i$th vNIC respectively, on a virtualized host. The static assignment for the capacity of each vNIC is:

$$C_{vNIC_1} = C_{vNIC_2} = \ldots = C_{vNIC_n} = C_1 \tag{1}$$

where $n$ is the total number of VMs (i.e., vNICs), and $C_1$ is a constant ($C_1 \leq C_{pNIC}$). To achieve better utilization when using multiplexing, we can have:

$$\sum_{i=1}^{n} C_{vNIC_i} \geq C_{pNIC} \tag{2}$$

This usually works well if all VMs are not busy at the same time. However, there may be heavy network congestion in cases where all VMs are busy performing network I/O processing. For example, suppose that there are 3 VMs on a virtualized host where $C_{pNIC} = C_1 = 1Gbps$. Then the total inbound traffic demand may increase to up to 3 Gbps while the pNIC capacity is only at 1 Gbps, resulting in poor

Manuscript received August 5, 2015.
Manuscript revised November 6, 2015.
Manuscript publicized November 30, 2015.

[†]The author is with Samsung Electronics, Suwon, Gyeonggi-do, Korea.
[††]The author is with the Department of Computer Science and Engineering, Korea University, Seoul, Korea.
[†††]The author is with the School of Computer Science and Information Engineering, The Catholic University of Korea, Gyeonggi-do, Korea.
a) E-mail: hjsuh@catholic.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2015EDL8173

network performance as a result of network congestion. If we use $C_{pNIC}/n$ for $C_1$, then network congestion could be always avoided since $\sum_{i=1}^{n} C_{vNIC_i} = C_{pNIC}$. However, the overall pNIC utilization would be reduced if there were any VM that does not use network resources.

## 3. Dynamic Rate Adjustment

To address the inefficiency of the static assignment, we developed a dynamic (inbound) rate adjustment scheme that works through a hypervisor. Our main design goals are as follows:

- Work-conserving: An increase in rate of one vNIC necessitates a decrease in the rate of another vNIC. In other words, we should maximize the total rate of vNICs without wasting pNIC capacity.
- Network congestion avoidance: The total amount of inbound traffic should not exceed the pNIC capacity to avoid the network congestion.
- Throughput fairness: Network throughput allocation between network connections should be fair across all VMs.

The first and second goals therefore cause the sum total of all the rates of the vNICs to be equal to the pNIC capacity. As a result, we adjust the rate of each vNIC to meet the third goal as follows:

$$C_{vNIC_i} = \frac{C_{pNIC} \times w_i}{\sum_{i=1}^{n} w_i} \tag{3}$$

where $w_i$ is the weight of the $i$th vNIC. In this scheme, the weight is the number of active network connections on each VM. Similarly to the previous data center studies [7]–[11], we mainly consider the partition/aggregate types of services that are delay-sensitive and generate many short-lived connections in a moment. To support this characteristic, we provide more bandwidth to the VMs that create more connections. Note that throughput-driven applications such as file transfers create a relatively small number of connections. The median number of concurrent throughput-driven connections per node is one in data centers [7], [8]. Therefore, they will be allocated less bandwidth by our scheme when competing with the partition/aggregate types of traffic. This is reasonable because they are not delay-sensitive, thus have a lower priority [8], [11]. In other words, the rates of the vNICs are adjusted in proportion to the number of connections so that the VM that has a higher bandwidth demand is assigned a higher rate relative to the others VMs.

Now we have to determine when the rate adjustment is performed. Since the bandwidth demand varies with time, the hypervisor should perform the rate adjustment according to (3) whenever there is a change in the bandwidth demand, i.e., when the number of active connections changes on any of the VMs. By doing so, we can always achieve throughput fairness between the network connections across all VMs. Note that the weight can be defined differently according to the operator's policy. For example, we may introduce a weight cap (i.e., the maximum number of connections) for VMs where the fairness of network resource usage is required. Finally, a minimum rate is assigned to the currently unused vNICs to ensure the minimum bandwidth required is available for future network connections.

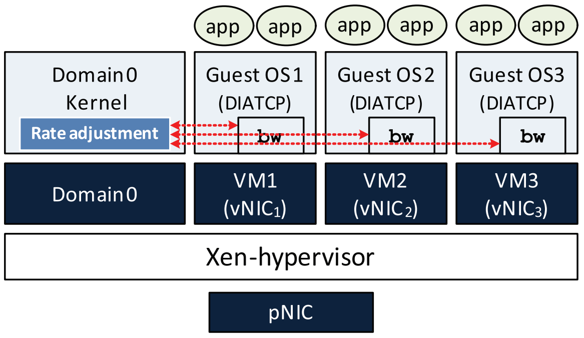

**Fig. 1** Architecture of the proposed rate adjustment scheme.

## 4. Implementation

We implement the *rate adjustment* module, in the domain0 kernel of the Xen-hypervisor [12] as shown in Fig. 1. As a data center protocol, DIATCP [8] is implemented in the TCP/IP stack of each guest OS (Linux kernel 2.6.38). The main role of DIATCP is to keep the rate of incoming traffic under the vNIC capacity by utilizing a TCP flow control scheme, as described in Sect. 1. Specifically, DIATCP controls its peers' sending window so that the aggregate incoming rate does not exceed the link capacity, i.e., its vNIC capacity, as follows:

$$\frac{\sum_{j=1}^{n} wnd_j \times MSS}{Average\_RTT} = C_{vNIC_i} \tag{4}$$

where $wnd_j$ is the TCP window size of the $j$th peer, advertised by DIATCP. MSS denotes the Maximum Segment Size. The value of $C_{vNIC_i}$ is stored in the internal variable, `bw`, so that DIATCP uses it for (4). The value can be modified by the rate adjustment module via the inter-domain communication (i.e., red dotted arrows). Whenever a guest OS opens/closes a network connection, it informs the rate adjustment module about the number of network connections of the guest OS. And then, the proposed scheme calculates the inbound rates of all vNICs and shares the calculated values with other guest OSs.

Note that, in our implementation, we mainly focus on TCP transmissions since it has been reported that 99.91% of traffic in data center networks is TCP [7]. Therefore, we allow such a negligible volume of UDP transmissions (i.e., $< 0.09\%$) to bypass the proposed scheme.

## 5. Evaluation

We evaluate our dynamic rate adjustment scheme on a small-scale cloud data center testbed, which consists of a Top-of-Rack (ToR) switch and 19 servers. All servers are connected to the ToR switch using 1 Gbps links. One of

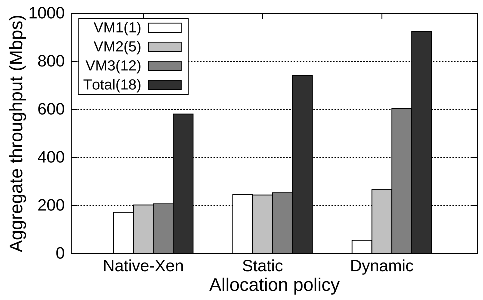

**Fig. 2** Aggregate throughput on each VM. The number in the bracket indicates the number of TCP connections (VM4 has no TCP connection).

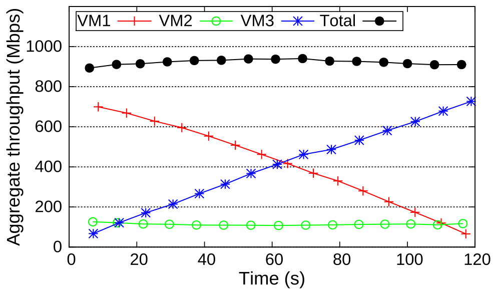

**Fig. 3** Dynamic rate adjustment varying the number of TCP connections on each VM.

those servers is a dedicated virtualized aggregator with 2 GB of main memory, running a domain0 host system of the Xen-hypervisor with four para-virtualized guest domains, VM1 to VM4, with 2 GB main memory each.

We establish 1, 5 and 12 TCP connections on VM1, VM2, and VM3, respectively. In each connection, the server sends a large amount of data to the aggregator. Figure 2 shows the aggregate throughput on each VM. In native Xen, all VMs share the pNIC capacity (1 Gbps) without any explicit vNIC capacity isolation. Since each VM tries to utilize the pNIC in a best-effort manner, packet drops frequently occur at the packet buffer of the ToR switch, causing throughput degradation with a total speed below 600 Mbps. The static policy avoids network congestion by assigning a constant capacity as high as $C_{pNIC}/n$ to each vNIC. However, it cannot achieve full pNIC utilization since the vNIC capacity assigned to VM4 of about 250 Mbps is unused (Fig. 2). In contrast, the dynamic rate adjustment of the proposed scheme fully utilizes the total pNIC capacity by assigning only a small rate to VM4. Furthermore, unlike in previous policies, the proposed scheme adjusts the rate of each vNIC in proportion to the number of TCP connections, achieving high throughput fairness among the connections of all VMs.

To verify how the proposed scheme dynamically performs the rate adjustment, we measure the aggregate throughput by (i) decreasing the number of TCP connections from 15 to 1 on VM1, (ii) increasing from 1 to 15 on VM3, and (iii) fixing the number to 2 on VM2, as shown in Fig. 3. The black line indicates the total throughput of all VMs. Figure 3 reveals that the pNIC is fully utilized with a total throughput always close to 1 Gbps. Hence, the hypervisor re-adjusts the rate of all VMs well whenever a network connection is created or closed on the VMs.

## 6. Conclusion

We proposed a new dynamic rate adjustment scheme for inbound data traffic on a virtualized host. The proposed scheme dynamically re-adjusts the rate of vNICs in collaboration with the TCP-level flow control scheme, so the amount of inbound data traffic of the VMs is proportionate to the given weight, e.g., the number of network connections. To evaluate the proposed scheme, we conducted experiments on our small-scale Xen-based cloud data center testbed. Through the experiments, we confirmed that the proposed scheme effectively achieves full pNIC utilization while performing dynamic rate adjustment of the vNICs based on their bandwidth demands.

## Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2057967) of Korea.

### References

[1] C.-H. Hong and C. Yoo, "Synchronization-Aware Virtual Machine Scheduling for Parallel Applications in Xen," IEICE Trans. Inf.& Syst, vol.E96-D, no.12, pp.2720–2723, Dec. 2013.
[2] S.K. Lee, H. Kim, J. Ahn, K.J. Sung, and J. Park, "Provisioning Service Differentiation for Virtualized Network Devices," Proc. International Conference on Networking and Services, May 2011.
[3] F. Dan, W. Xiaojing, Z. Wei, T. Wei, and L. Jingning, "vSuit: QoS-oriented scheduler in network virtualization," Proc. International Conference on Advanced Information Networking and Applications Workshops, pp.423–428, March 2012.
[4] C. Guo, G. Lu, H.J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees," Proc. ACM CoNEXT, 2010.
[5] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the Data Center Network," Proc. USENIX NSDI, 2011.
[6] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, C. Kim, and A. Greenberg, "EyeQ: Practical Network Performance Isolation at the Edge," Proc. USENIX NSDI, 2013.
[7] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," Proc. ACM SIGCOMM, 2010.
[8] J. Hwang, J. Yoo, and N. Choi, "Deadline and Incast Aware TCP for Cloud Data Center Networks," Computer Networks, vol.68, pp.20–34, Aug. 2014.
[9] J. Hwang, J. Yoo, and N. Choi, "IA-TCP: A Rate Based Incast-Avoidance Algorithm for TCP in Data Center Networks," Proc. IEEE ICC, pp.1292–1296, June 2012.
[10] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast Congestion Control for TCP in Data-Center Networks," IEEE/ACM Trans. Netw., vol.21, no.2, pp.345–358, 2013.
[11] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better Never than Late: Meeting Deadlines in Datacenter Networks," Proc. ACM SIGCOMM, 2011.
[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," Proc. ACM SOSP, Oct. 2003.