Living on the Edge: Monitoring Network Flows at the Edge in Cloud Data Centers

Vijay Mann * , Anilkumar Vishnoi * , Sarvesh Bidkar †
* IBM Research - India
Email:{vijaymann,avishnoi}@in.ibm.com
†Indian Institute of Technology, Mumbai, India
Email:sarvesh@cse.iitb.ac.in

***Abstract*—Scalable network wide flow monitoring has remained a distant dream because of the strain it puts on network router resources. Recent proposals have advocated the use of coordinated sampling or host based flow monitoring to enable a scalable network wide monitoring service. As most hosts in data centers get virtualized with the emergence of the cloud, the hypervisor on a virtualized host adds another network layer in the form of a vSwitch (virtual switch). The vSwitch now forms the new edge of the network.**

In this paper, we explore the implications of enabling network wide flow monitoring inside virtual switches in the hosts. Monitoring of network flows inside the server vSwitch can enable scalability due to its distributed nature. However, assumptions that held true for flow monitoring inside a physical switch need to be revisited since vSwitches are usually not limited by the same level of resource constraints that exist for physical switches and routers. On the other hand, vSwitches do not implement flow monitoring in hardware, as it is done in some physical switches. We present the design and implementation of EMC2 - a scalable network wide monitoring service for cloud data centers. We also conduct an extensive evaluation of various switch based flow monitoring techniques and share our findings. Our results indicate that while layer-3 flow monitoring protocols such as NetFlow can give a very good network coverage without using too many resources, protocols that sample packet headers (such as sFlow) need to be carefully configured. A badly configured sFlow vSwitch can degrade application network throughput by up to 17% and can also choke the management network by generating monitoring data at a very high rate.

Keywords: Flow monitoring, sampling, NetFlow, sFlow, vswitch

I. Introduction

Network monitoring is an important aspect of network management. As data center networks move to a flexible and programmable control plane provided by software defined networking technologies such as OpenFlow [1], the importance of network wide traffic monitoring increases significantly. Network monitoring provides key inputs to the control plane, based on which the control plane can take actions and implement traffic engineering [2] [3] [4]. OpenFlow itself provides basic statistics in the form of flow counters for each OpenFlow rule. This adds to the list of several other protocols that have been proposed over the years for network traffic measurement. These include IETF RMON (Remote Monitoring) [5] which is based on SNMP as well as flow monitoring protocols such as sFlow [6] and NetFlow [7] that can measure bytes and packets for each flow (uniquely identified through a source and destination IP address) in the network.

978-1-4673-5494-3/13/$31.00

To achieve network wide monitoring of traffic, any monitoring approach has to be scalable and largely non-intrusive. Packet sampling based approaches have been designed to increase scalability. Similarly, to reduce network load of fetching network statistics, push based approaches have been preferred over pull or polling based approaches. For these reasons, sampling based flow monitoring protocols have become the de facto industry standard and are supported on most commercial network equipment because of their ease of use and scalability. However, even sampling based approaches suffer from problems since some of them implement uniform packet or randomized packet sampling but not flow sampling. Choosing a low sampling rate may result in poor network coverage and result in large approximation errors in reporting small flows or even missing out on small flows entirely. On the other hand, selecting a high sampling frequency can drain the switch resources. Most of the earlier work [8] [9] [10] has focused on increasing the accuracy of single router measurements through incremental solutions. Recent proposals such as [11] and [12] have proposed changes to sampling based flow monitoring with the objective of increasing its scalability and fidelity for network wide monitoring. For example, Coordinated Sampling [11] proposes flow based sampling and a hash based division of network flows and monitoring responsibilities among the routers to achieve network wide monitoring goals while respecting router resource constraints. Other recent works such as [13] and [14] propose monitoring of network traffic at the hosts. Host level monitoring of network flows enables scalability as it achieves collaborative monitoring by design since each host is responsible for monitoring only those network flows that either originate from it or end in it. However, most earlier approaches have employed custom modifications to the networking stack in the host OS. This reduces the practical feasibility of such solutions.

With the emergence of the cloud, and wide spread adoption of server virtualization in data centers, most hosts in data centers are being virtualized. Each hypervisor on a virtualized host adds another network layer in the form of a "vswitch" (virtual switch), which form the new "edge" of the network. Realizing the importance of network-wide monitoring of flows, hypervisor vendors are increasingly adding

support for network monitoring within the hypervisor virtual switch. For example, VMware vSphere supports NetFlow in vSphere 5.0 onwards [15]. OpenVswitch [16] (OVS) that has implementations for both KVM [17] and Xen [18] supports both sFlow and NetFlow. Hyper-v [19] has in-built support for sFlow monitoring. The support for flow monitoring within the vSwitch can enable a scalable network wide monitoring service for cloud data centers.

In this paper, we explore the implications of enabling network wide flow monitoring inside virtual switches in the hosts in cloud data centers. Monitoring of network flows inside the server vSwitch presents an interesting design point. However, some of the assumptions that held true for flow monitoring inside a physical switch need to be revisited since vSwitches are usually not limited by the same level of CPU and memory resource constraints that exist in the physical switches and routers. On the other hand, vSwitches do not implement flow monitoring in hardware, as it is done in some physical switches (mainly for sFlow). Furthermore, it may be essential to implement flow monitoring inside the vSwitches to capture inter-VM flows for VMs that are located on the same physical host, since these flows never get out to the physical network. We make the following contributions in this paper:

- We formulate the requirements for a network traffic monitoring service in a cloud data center.
- We present the design and implementation of EMC2 (Edge Monitoring and Collector for Cloud) - a scalable network-wide monitoring service for cloud data centers.
- We conduct an extensive evaluation of various vSwitch based flow monitoring techniques on different platforms and share our findings. Our results indicate that while layer-3 flow monitoring protocols such as NetFlow can give a very good network coverage without using too many resources, protocols that sample packet headers (such as sFlow) need to be carefully configured. A badly configured sFlow vSwitch can degrade application network throughput by up to 17% and can also choke the management network by generating monitoring data at a very high rate (600 Mbps for a 10,000 server data center). These results indicate that sFlow implementations that were designed to be implemented in hardware on physical switches, may need to be redesigned for vswitch based software implementations.

The rest of this paper is organized as follows. Section II presents some background and an overview of related research. In Section III, we present key requirements for a network traffic monitoring service in a cloud data center. We present the design and implementation of EMC2 in Section IV. Section V describes our experimental results. We summarize our findings and give an overview of our current and future work in Section VI.

II. Background and Related Work

A. Background

Flow monitoring inside network routers and switches using NetFlow and sFlow

NetFlow [7] has been the de-facto standard for flow monitoring for many years. NetFlow enables monitoring of network flows at layer 3. The original NetFlow protocol enabled monitoring of every flow since each packet was monitored. In order to preserve router resources, sampling based NetFlow (referred to as Sampled NetFlow [9] was proposed. This was further improved upon by Random Sampled NetFlow [20]. Currently NetFlow is supported on a wide variety of routers and switches such as those implemented by Misc, Juniper (it supports jFlow - a NetFlow variant), VMware's vSphere distributed vSwitch [15] and OpenVswitch [16].

sFlow [6] is an emerging standard supported by many switch vendors such as IBM, HP, and OpenVswitch [16]. sFlow samples packet based on the specified sampling rate and for each sampled packet, it reads in a specified number of header bytes. Typically a default value of 128 packet header bytes is read [21]. These header bytes enable sFlow to report statistics on layer 2 traffic as well as any other statistic from layer 2 to layer 7. However, these header bytes can cause an enormous strain on the router resources and hence, most switches restrict the users to a default sampling rate of 256 or more [21]. This can cause several approximation errors in reporting of network flows as well as traffic volumes.

B. Related Research

Related research in network monitoring can be broadly classified into the following categories.

Flow monitoring inside hosts using socket buffers:

Authors in [13] propose a scheme in which the end-host marks a flow as an elephant flow based on the level of socket buffers inside the host. The reporting of these flows is in-band using the DSCP [22] field in the IP header of the packet. In [14], the authors use a similar host based flow monitoring approach based on socket buffers to take routing decisions based on the traffic matrix of ToR-to-ToR switches. In this approach they combine the host-to-host flow data from all the hosts in a rack into a designated host and convert it into ToR-ToR flow data. This aggregated flow data is then reported to a centralized controller which computes the optimal paths based on the current traffic matrix and configures these paths into the network devices. Flow monitoring inside hosts enables scalability as it achieves collaborative monitoring by design since each host is responsible for monitoring only those network flows that either originate from it or end in it. However, most earlier approaches have employed custom modifications to the networking stack in the host OS. This reduces the practical feasibility of such solutions.

Improvements to flow monitoring inside routers and switches: The works in [8] [11] [12] [10] describe implementation of monitoring agents in physical routers and switches in the network and present techniques that can be used to minimize CPU overhead and memory requirements while getting accurate estimates about network traffic and flows.

[11] proposes a network wide, coordinated monitoring approach using flow based sampling. The monitoring nodes in the network are given a list of flows that they need to monitor based on disjoint hash ranges. This ensures maximum flow coverage while eliminating redundant flow reporting in the network. Their architecture is based on a feedback mechanism where, they extract the traffic matrix from the reported flow statistics, and then calculate and distribute the sampling manifests to monitoring nodes based on an optimization formulation which satisfies monitoring resource constraints and ensures maximum flow coverage.

In a follow-up work to [11], that considers a minimalistic approach for flow monitoring, the authors propose an architecture for supporting a wide range of network management applications [12]. The sample-and-hold (HS) and flow sampling (FS) primitives described in the paper can work well within the router resource constraints and provide required monitoring information.

The work in [8] presents the limitations of an early implementation of sampling based NetFlow and proposes solutions to improve the performance of NetFlow in terms of the accuracy of traffic estimation. The techniques used are adaptive sampling rate which adjusts the sampling rate of NetFlow based on the traffic mix observed on a given port. The technique of renormalization of flow entries reduces sampling overhead and saves on reporting bandwidth as the sampling rate increases. The problem of time bin mismatch between the management application and the NetFlow agent in the router is solved simply by dividing the NetFlow operation in the router such that the time bins used by management applications are exact multiples of time bins used by these operations.

The work in [9] presents an analysis of sampling based NetFlow approach and establishes the bounds on number of samples required to get a certain accuracy in estimation. The methods of systematic sampling and random sampling are also analyzed for their limitation of accuracy. The authors of [10] propose a light-weight NetFlow agent implementation in the ToR switches and a collector system which collects the exported records and presents them to a management application after converting them into a compatible format. The proposed solution employs a sampling based technique similar to sFlow [6] and requires changes in the switch implementation as well as a converter to support existing NetFlow based management applications.

There has been very little work that explores scalability of flow monitoring inside virtual switches (or vSwitches). We believe that a lot of existing work on improving flow monitoring inside physical switches can be revisited in this new context and can be applied with appropriate modifications.

III. Key Requirements

In this section, we present key requirements for a network traffic monitoring service for the cloud that have been formulated based on our interactions with various data center system administrators and developers of cloud based systems as part of our ongoing work on network-aware VM management in cloud data centers [4] [3].

- **R1.** The most important requirement for a network traffic monitoring service (like any other monitoring service) is that it should be largely non-intrusive and should not degrade the performance of the monitored system in any significant manner (no more than 5% degradation).
- **R2.** Most cloud scenarios such as traffic engineering [2] [23] and network-aware VM placement [3] [4] (that places highly communicating VMs close to each other in terms of network distance) require that the network traffic monitoring service should give accurate estimation of elephant (large) flows in the network even if it misses out on some mice (small) flows. This is in contrast to current requirements from ISPs which typically require the monitoring service to provide good network coverage (that includes both elephant and mice flows).
- **R3.** Most cloud scenarios are mainly concerned about gathering monitoring data at layer-3 and not so much about layer-2 data. This is because, it is easier to act (e.g. for traffic engineering) upon layer-3 monitoring data through a programmable control plane such as that provided by OpenFlow [1]. Furthermore, most cloud providers do not want to look into their tenants' packet headers (e.g. to find out what protocols are being used the most) either because this violates their contract agreements or because they have little use of such data.
- **R4.** A network traffic monitoring service for the cloud should ensure that it works across a heterogeneous mix of host platforms and physical switches.
- **R5.** A network traffic monitoring service should export its data through rich interfaces that can be easily plugged into various other systems for control plane action, reporting and anomaly detection.
- **R6.** A single network traffic monitoring service should provide information about various aspects of the network such as network flow volumes, network topology and network link utilizations. Typically, these are reported through different services and products in current data centers due to lack of standardization across these aspects.

IV. Design and Implementation

In this section we describe our design and implementation of EMC2 (Edge Monitoring and Collection for Cloud). Figure 1 gives an overview of our architecture. EMC2 relies on flow monitoring capabilities within the virtual switch on each hypervisor. Hypervisors today support either NetFlow (VMware vSphere 5.0 onwards) or sFlow (Hyper-v) or both (OpenVswitch). EMC2 assumes that a cloud data center will comprise of a mix of host platforms and vSwitches (as required by **R4** in section III). It also currently focuses only on monitoring flows at layer-3 (as required by **R3** in section III).

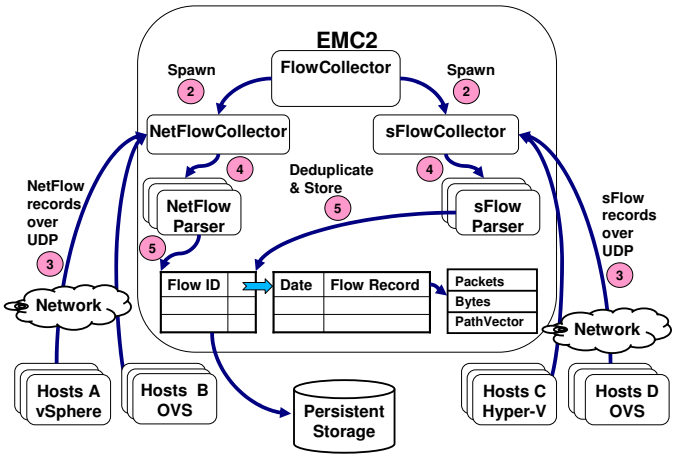

Fig. 1. EMC2 Architecture

A. Architecture

EMC2 has a fully multi-threaded architecture and has been developed in Java. It spawns two threads - one for accepting NetFlow datagrams (NetFlowCollector) and another for accepting sFlow datagrams (sFlowCollector). Each NetFlow or sFlow flow datagram contains entries for multiple flows that were observed during the last monitoring interval. The collector threads, upon receiving a NetFlow or sFlow datagram, spawn a dedicated parser thread (NetFlowParser or sFlowParser). EMC2 also maintains an in-memory flow table for storing flow records. The in-memory flow-records are later persisted onto a persistent storage (such as a database or flat files - we currently use flat files). Each flow is identified using a flow ID formed by concatenating its layer-3 source and destination addresses. This forms a key into the in-memory flow table. The in-memory flow table is a 2-level Hashtable where the primary key is the flow ID and the value is another table in which the timestamp of a particular flow record is used as the key and the flow record is the value. A flow record in EMC2 comprises of the number of packets, number of bytes and an optional path vector corresponding to a particular update (at a given time instance) for a given flow ID. The path vector stores the path in the network taken by a particular flow. This can be easily determined if all the switches in the data center network (and not just vSwitches), have been configured to send NetFlow or sFlow datagrams. The parser threads will detect multiple updates from different switches for the same flow and use the time difference between consecutive updates to determine if these updates correspond to the same set of packets. Even though, both sFlow and NetFlow datagrams contain timestamp values, EMC2 uses its own timestamps (which it records upon receiving a NetFlow or sFlow datagram). The parser threads perform two important tasks:

- **Deduplication,** and
- **Data rate prediction in presence of sampling**.

These are described in more detail in the following subsections.

B. Deduplication

Deduplication refers to the checks that prevent duplicate flow records to be added to the in-memory flow table. Duplicate flow records refer to the same flow being reported by multiple vSwitches. For example, every flow between Host A and Host B, will be reported by the vSwitch in Host A as well as the vSwitch in Host B (and by other hardware switches in the network if they have been configured for sFlow or NetFlow). Parser threads follow simple heuristics to prevent duplicate flow records. If the flow ID does not exist in the in-memory flow table, it is assumed to be a new flow that has not been observed before. If the flow ID exists in the in-memory table, it can either be a duplicate update (from another vSwitch or hardware switch in the network) or it can be a new update for the same flow for the next monitoring interval. A duplicate update will have a different exporter address (exporter address refers to the vSwitch or physical switch that is sending NetFlow or sFlow data - this information is included in both NetFlow and sFlow datagrams). EMC2 ensures that statistics about each flow are maintained using updates from same exporter address. Updates about the same flow from other exporter addresses are just used to update the path vector.

EMC2 also supports an optional "loss-tolerant" mode where it tries to account for the fact that some NetFlow or sFlow updates might be lost in the network (since they are sent over UDP). In this mode, instead of relying on the exporter address, it relies on timestamp values to detect duplicate flows. If the same flow gets monitored by NetFlow and sFlow (e.g. this can happen if the source host has Hyper-V and the destination host has VMware vSphere hypervisor), it gives preference to NetFlow (with sampling rate=0, which indicates that every NetFlow record is being sent to the collector) since it is not likely to suffer from sampling errors. It follows the following strategy:

- If a NetFlow record already exists for a given flow ID, sFlow records for that flow are not used (for determining layer 3 stats) since it indicates that the given flow ID is being monitored through a NetFlow capable vswitch which does not suffer from sampling errors. For NetFlow records, it uses the timestamp values to detect duplicates. If the difference between the timestamp value for the current flow record and the last observed flow record for the given flow ID is more than the NetFlow export timeout (which is a user configurable parameter), it is assumed to be a new update, else it is assumed to be a duplicate update from a different exporter device.
- If a NetFlow record does not exist for a given flow ID, sFlow samples for that flow are accumulated from different exporter IDs and the cumulative count of these sFlow samples in a given time interval is used to predict the data rate for that flow. If the sFlow sample comes first

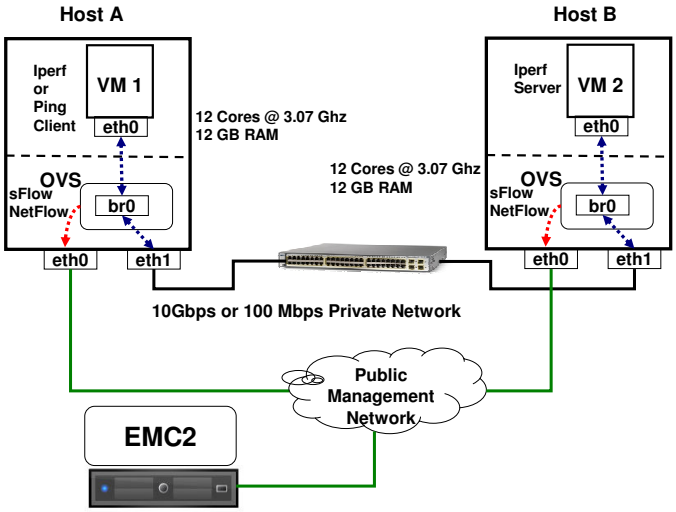

Fig. 2. Evaluation Testbed for OpenVswitch (OVS) - VMware testbed is similar except that the private network has 1 Gbps and the servers have 12 Intel cores at 3.47 GHz each

for a flow ID, followed by a NetFlow packet (with sampling rate=0), sFlow packets are discarded and NetFlow samples are given preference (since they do not involve sampling errors).

C. Data rate prediction in presence of sampling

In presence of sampling, the data rate for each flow has to be accurately predicted. Sampling rate is specified in each sFlow and NetFlow datagram. Currently, EMC2 just multiplies the sampling rate with the length of the packet specified in the layer 3 header. This ensures that the numbers reported by sFlow based vSwitches are in agreement with those reported by NetFlow based vSwitches. There is still a possibility of under-reporting the total number of bytes or of missing out an entire flow. In order to reduce sampling errors, it detects the sampling rate from the sFlow and NetFlow datagrams and for flow IDs that report low sampling rates, it accumulates samples from different exported devices for the same flow ID and then divides the total samples by the number of exporter devices that reported that sample. This reduces the probability of missing out on small flows.

V. Feasibility Evaluation

In this section, we present an extensive evaluation of various vSwitch based flow monitoring techniques on different platforms. In particular, we evaluate the NetFlow and sFlow support in OpenVswitch (OVS) and the NetFlow support in VMware vSphere 5.1 [15], since OVS and vSphere are the two most widely used vSwitch implmentations that we have come across. We used two distinct but similar testbeds - one with OVS and another one with VMware vSphere for ease of use and to avoid re-installation of the hypervisor. In all the experiments EMC2 service was used to configure the vSwitches and to collect monitoring data from the vSwitches.

We performed experiments to evaluate the effect of sampling frequency on:

- server CPU utilization
- application's network throughput
- application's network latency
- the monitoring data generated

Note that OVS supports setting the sampling rate for sFlow to any value. Most physical switches restrict the users from setting the sampling rate for sFlow to anything below 256 [21], to protect switch resources. We varied the sampling rate from 800 (1 out of 800 packets is sampled, resulting in a sampling rate of 0.00125) to 1 (every packet is sampled). OVS does not support setting of sampling rate for NetFlow.

A. Evaluation Testbed

Figure 2 shows our experimental testbed for OpenVswitch (OVS). It comprises of two large multicore physical machines (Host A and Host B) - each with 12 Intel Xeon cores at 3.07 GHz each and with 12 GB RAM. Both these machines run Ubuntu 12.04 LTS OS and the latest version of KVM [17] as the hypervisor. Each machine has two network interface cards (NIC). One NIC is connected to a public network that we use as the management network for configuring vSwitches as well as for exporting NetFlow and sFlow data. The second NIC is used to connect the two machines in a private network through a single switch. OVS is configured on each machine to use this second NIC as the uplink through a bridge (shown as "br0" in Figure 2). Each machine hosts 1 VM and has a virtual NIC that is mapped to the bridge "br0". Each VM is allocated 1 GB of RAM and only 1 vCPU. This leaves the server largely free in terms of CPU (only 1 out 12 cores is utilized, and the rest is free for the hypervisor). We conduct two sets of experiments - one in which the private network has 100 Mbps capacity and another in which the private network has 10 Gbps capacity. This is achieved by replacing the single switch with a higher capacity switch for 10 Gbps. The experimental testbed for VMware vSphere is similar to the OVS testbed in all aspects except 3 main differences:

- the private network has 1 Gbps capacity;
- that servers have 12 Intel Xeon cores each at 3.47 GHz; and
- the servers host several other VMs that were shut down for our experiments

Note that vSphere supports sampling in its NetFlow implementation as opposed to OVS that does not support sampling for NetFlow. We could not conduct VMware vSphere experiments with different link speeds since that testbed was not our direct control and we had to conduct all experiments at 1 Gbps link capacity.

B. Effect of Sampling Rate on Server CPU Utilization

In these experiments, we drive network traffic between Host A and Host B after enabling flow monitoring on the OVS (on Iperf client machine - Host A), and measure average server CPU utilization and hypervisor process CPU utilization.

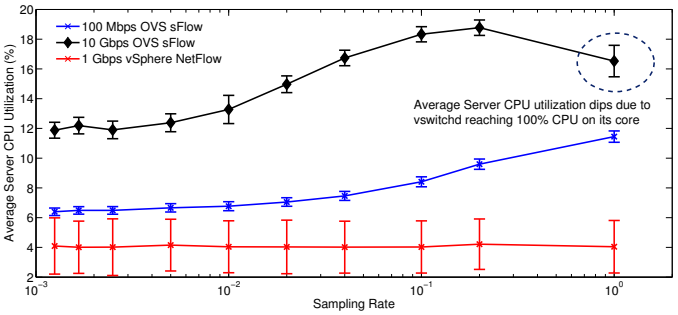

Fig. 3. Average server CPU utilization during Iperf experiments: CPU utilization shows a steep increase as the sampling rate is increased from 0.00125 to 1 for sFlow on OVS. It dips for sampling rate=1 in the 10Gbps network as ovs-vswitchd (the vSwitch process) reaches 100% CPU Utilization on its core. CPU utilization remains steady with vSphere sampled Netflow.

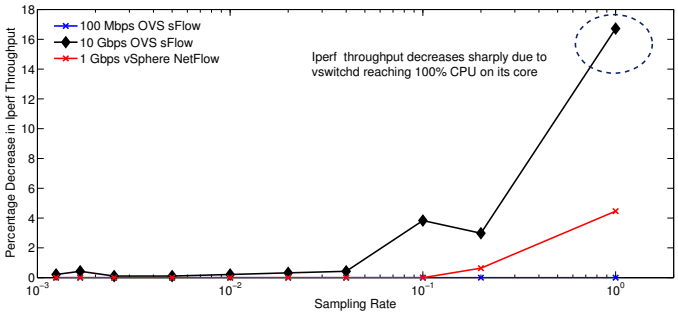

Fig. 4. Iperf throughput shows a sharp decrease by around 16% in the 10 Gbps network for sampling rate=1 as ovs-vswitchd process consumes close to 100% of its CPU at high sampling rates and becomes a bottleneck

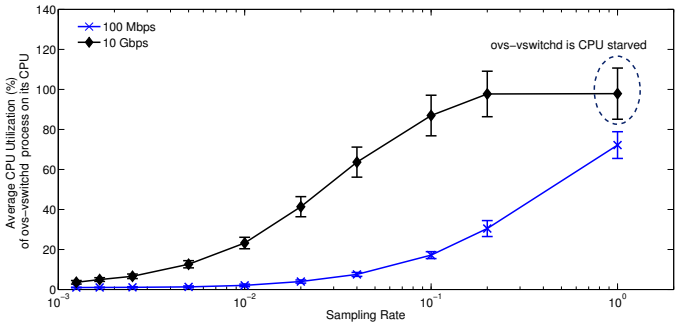

Fig. 5. CPU Utilization of ovs-vswitchd process during Iperf Experiments: ovs-vswitch process consumes close to 100% of its CPU at high sampling rates and becomes a bottleneck

We use nmon [24] to measure server[1] CPU utilization and Iperf [25] to drive network traffic between the two VMs. In this set of experiments, VM2 on Host B runs an Iperf server, and VM1 on Host A runs an Iperf client. Each test runs for a period of 5 minutes and we repeat the test for various sampling rates (0.00125 to 1) with sFlow in the 100 Mbps network as well as in the 10 Gbps network. The default value of 128 header bytes to be sampled was used. For vSphere runs, we varied the NetFlow sampling rate in a similar manner. Average server CPU utilization of Host A (that runs Iperf Client) for each of the 5 minute runs is shown in Figure 3. For sFlow runs on OVS, the following trend can be observed: As sampling rate is increased from 0.01 (1 out of 100 packets is sampled), the average CPU utilization increases steadily and almost becomes double of its original value (from 6% at 0.00125 to 12% at 1) in the 100 Mbps network. The trend is similar in the 10 Gbps network, with two major differences:

- Average CPU utilization is much higher even at low sampling rates (0.00125) - this is because at high data transfer speeds, the sampling overhead increases for the same sampling rate.
- Average CPU utilization increases steadily till sampling rate 0.2 (1 out of 5 packets sampled), but then decreases sharply as the sampling rate is increased to 1(every packet is sampled).

CPU utilization of the server with sampled NetFlow on vSphere, in contrast, does not show any variation in average CPU utilization as sampling rate is increased. Note that, the standard deviation is higher in this case, since this host has several VMs (as compared to just 1 VM on each OVS host). While, all these VMs were shutdown for our experiments, still their presence introduces some variance.

The percentage degradation in throughput or available bandwidth reported by Iperf (over a no monitoring baseline case) for these experiments is shown in Figure 4. Iperf throughput does not show any degradation in the 100 Mbps network. However, in the 10 Gbps network, for sFlow on OVS, Iperf throughput shows mild degradation of around 4% at sampling rates 0.1 (1 out of 10 packets sampled) and 0.2 (1 out of 5 packets sampled) and significant degradation of around 17% at sampling rate of 1. Iperf throughput for vSphere sampled NetFlow configuration shows a minor degradation of around 4% when every packet is sampled (sampling rate=1).

A closer look at the per process CPU utilization on OVS Host A reveals interesting facts about the average CPU utilization of OVS process responsible for the vSwitch (ovs-vswitchd) (refer Figure 5). As sampling rate is increased, the average CPU utilization of ovs-vswitchd process increases steadily and reaches around 20% of its CPU core at around 0.1 sampling rate (1 out of 10 packets) in the 100 Mbps network and at around 0.01 sampling rate (1 out of 100 packets) in the 10 Gbps network. Thereafter, it increases sharply in both cases and becomes almost 100% at 0.2 sampling rate (1 out of 5 packets) in the 10 Gbps network. Any further increase in the sampling rate in the 10 Gbps network results in a sharp degradation of network throughput since ovs-vswitchd gets CPU starved. This explains the sharp decline in Iperf throughput at sampling rate 1 in the 10 Gbps network.

C. Comparison of NetFlow and sFlow monitoring on OVS

In this section we compare the server CPU utilization for NetFlow and sFlow monitoring on OVS. The experimental

[1]We use the terms server and host, interchangeably in this section to refer to a physical machine

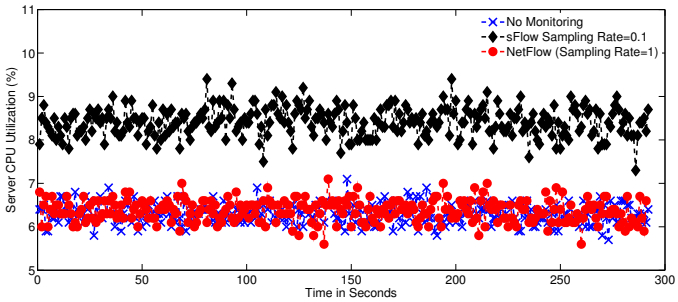

Fig. 6. Average server CPU utilization during Iperf experiments in the 100 Mbps network: Average Server CPU utilization with NetFlow is almost the same as the CPU utilization with no monitoring. This implies that NetFlow can give network wide coverage of flows without incurring any cost on the OVS

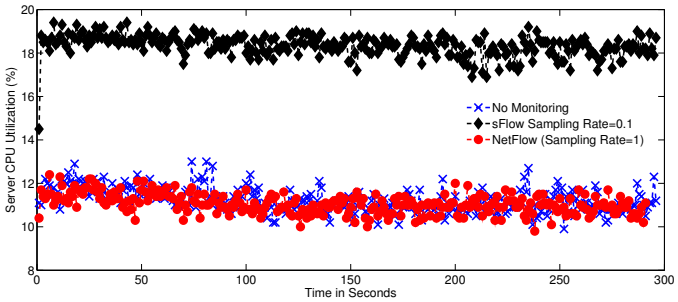

Fig. 7. Average server CPU utilization during Iperf experiments in the 10 Gbps network: Average Server CPU utilization with NetFlow is almost the same as the CPU utilization with no monitoring.

setup remains the same as in the previous set of experiments. We repeat the Iperf runs after enabling NetFlow based flow monitoring on OVS (on Iperf Client machine - Host A). The CPU utilization of Host A for the 5 minute runs in the 100 Mbps network is shown in Figure 6 and in the 10 Gbps network is shown in Figure 7. We compare NetFlow with sFlow (with sampling rate=0.1) as well as with a baseline no monitoring scenario. Two interesting observations can be made from these two graphs:

- The server CPU utilization for NetFlow is almost the same as the baseline server CPU utilization that has no monitoring enabled. This has interesting ramifications. This implies that at least on the OVS, one can have a full network coverage at layer-3 without incurring any significant overheads. On vSphere, the sampled NetFlow implementation can also enable full network coverage with very minimal overheads (less than 4%). As we mentioned in Section III, a lot of cloud scenarios on traffic engineering and network-aware VM placement, only need layer-3 information and NetFlow on OVS can easily provide this information without incurring significant overhead (which is typically not the case with network wide flow monitoring).
- sFlow based monitoring induces significant CPU overhead and this is mainly due to the CPU consumed by the ovs-vswitchd process (refer Figure 8). ovs-vswitchd process can consume up to 80% of its CPU core to support sFlow at a sampling rate of 0.1 (1 out of 10 packets). In contrast, NetFlow support on OVS can be enabled at almost no extra cost (ovs-vswitch consumes less than 2% of its CPU core to support NetFlow).

D. Effect of Sampling Rate on Monitoring Data Generated

In this section we compare the monitoring data generated by NetFlow and sFlow monitoring on OVS. The experimental setup remains the same as in the previous set of experiments. We repeat the Iperf runs after enabling NetFlow or sFlow based flow monitoring on OVS (on Iperf Client machine - Host A) and record the network data on the management network.

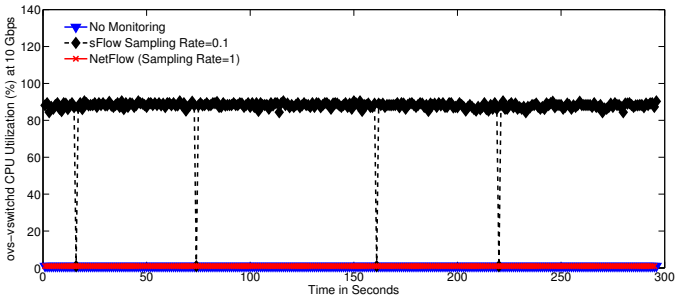

Fig. 8. CPU Utilization of ovs-vswitchd process during Iperf experiments in the 10 Gbps network: ovs-vswitchd process consumes significant CPU (around 80% on its core) for sFlow monitoring (at sampling rate=0.1). On the other hand, it consumes very little CPU (less than 2% on its core) for supporting NetFlow based monitoring which is almost the same as if no monitoring was enabled on the OVS.

The results are shown in Figure 9 for NetFlow and sFlow with various sampling rates. Recall that NetFlow on OVS does not support sampling rates, and the results shown are for the default configuration which is equivalent to a sampling rate of 1.

It is interesting to note that sFlow can generate nearly 60 Kbps worth of monitoring data per server (in the 10 Gbps network) for sampling rates of 0.1 (1 out of 10 packets sampled) and beyond. This corresponds to 60 Mbps monitoring data in a 1000 server data center, and 600 Mbps monitoring data in a 10,000 server data center. This can be considered too high for a monitoring service. On the other hand, NetFlow on OVS does not generate more than 0.05 Kbps per server (50 Kbps for 1000 servers and 500 Kbps for 10000 servers). This can be a big advantage. However, each NetFlow datagram comprises of an update with all flows that have been observed in the last monitoring interval. This implies that as number of flows observed on each host increase,the size of the NetFlow datagram will also increase. Hence, the monitoring data for NetFlow would have been higher if we had more flows on the host. Assuming 100 flows on each host on average and applying a linear extrapolation, this corresponds to around 50 Mbps monitoring date rate in a 10,000 server data center. This is still lower, by almost a factor of 10, than the monitoring

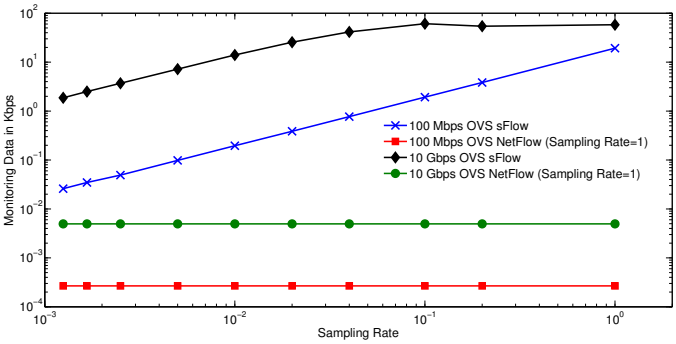

Fig. 9. sFlow generates around 60 Kbps monitoring data per server - this is equivalent to around 600 Mbps worth of monitoring data in a 10,000 server data center.

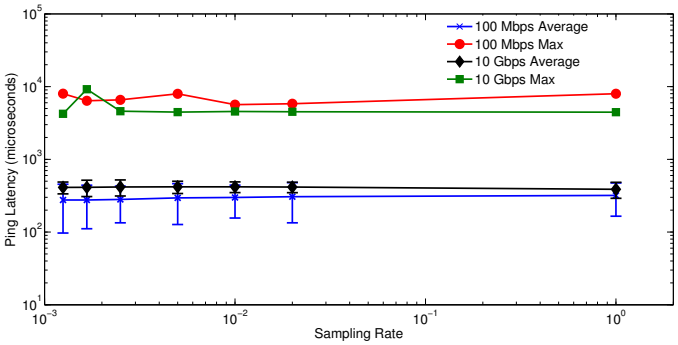

Fig. 10. Average ping latency does not get affected much by variation in sampling rate - slight increase in latency can be noted in the 100 Mbps network with increase in sampling rate

data exported by sFlow at a sampling rate of 0.1.

Networking monitoring using sFlow at high sampling rates within the vSwitch will require a distributed architecture like the ones proposed in [14] [13]. We are working on a distributed version of EMC2 in which multiple collector instances can be deployed for different parts of the network and they can report their aggregate statistics to a master collector.

E. Effect of Sampling Rate on Application Latency:

In these experiments, we measure the effect of flow monitoring at various sampling rates on application latency. To validate this, VM1 on Host A pings VM2 on Host B with a very low interval (of 100 milliseconds) and we continue this experiment for around 5 minutes. We repeat the same experiment after enabling sFlow monitoring in the OVS and vary the sampling frequency from 0.00125 (1 out of 800 packets is sampled) to 1 (every packet is sampled). Average and maximum ping latencies are shown in Figure 10 for 100 Mbps and 10 Gbps network. Both average and max latencies do not seem to be affected much by the sFlow sampling rate. This is mainly because even with a small interval of 0.01 millisecond, a ping process does not generate enough data to stress the ovs-vswitchd process in the vSwitch.

F. Discussion

Our results can be summarized as follows:

- Using NetFlow, one can have a full network coverage at layer-3 without incurring any significant overheads. On OVS, the NetFlow overheads are less than 1% whereas on vSphere, the sampled NetFlow implementation can also enable full network coverage with very minimal overheads (less than 4%). This satisfies our most important requirement **R1** from section III.
- Our results indicate that sFlow can degrade application network throughput at high sampling rates. It can also result in monitoring date being exported at very high data rate (around 600 Mbps) in the management network thereby choking the managing network. This is not entirely surprising, since sFlow is a packet sampling technique whereas NetFlow is a flow sampling technique. sFlow samples every nth packet (where n is determined by the sampling frequency). For each sampled packet, it reads in x bytes of the header (where x is usually 128 bytes by default [21]) and sends those x bytes to the collector. This kind of packet processing (done inside the ovs-vswitchd process for OVS) can put too much strain on the CPU resources of a server. Some physical switches implement sFlow in hardware and this makes them perform much better with sFlow. Furthermore, most switches do not allow users to specify a sampling rate of less than 256 [21]. Still, it is interesting to note that a server can handle a sFlow sampling rate of around 100 reasonably well (with 20% CPU utilization on one of the cores where ovs-vswitchd runs) on a 10 Gbps network. This sampling rate is much higher than what is usually recommended as the default sampling rates for switches [26]. Typically a sampling rate of around 2000 is recommended for 10 Gbps switch, since the performance requirements of a switch are much stricter than that of a large multi-core server.

Even though, it might seem that more experiments with larger number of flows should be conducted, we believe that our results are representative and any experiments with larger number of flows will have similar results. This is because sFlow samples packets instead of flows and Iperf sends data at the maximum rate at which the network allows. Hence, it does not matter whether the packets being sampled by the sFlow agent inside the vSwitch belong to a single flow or multiple flows.

VI. Conclusion and Future Work

In this paper, we explored the implications of enabling network wide flow monitoring inside virtual switches in the hosts in cloud data centers. We formulated the requirements for a network traffic monitoring service in a cloud data center. We also presented the design and implementation of EMC2 (Edge Monitoring and Collector for Cloud) - a scalable network-wide monitoring service for cloud data centers. We conducted an extensive evaluation of various vSwitch based flow monitoring techniques on different platforms and

shared our findings. Our results indicate that while layer-3 flow monitoring protocols such as NetFlow can give a very good network coverage without using too many resources, protocols that sample packet headers (such as sFlow) need to be carefully configured. A badly configured sFlow vSwitch can degrade application network throughput by up to 17% and can also choke the management network by generating monitoring data at a very high rate (up to 600 Mbps in a 10,000 server data center). These results indicate that sFlow implementations that were designed to be implemented in hardware on physical switches, may need to be redesigned for vSwitch based software implementations.

We believe that a lot of existing work on improving flow monitoring in physical switches can be revisited in this new context of flow monitoring inside vSwitches and can be applied with appropriate modifications. We are currently working on a number of enhancements to EMC2 such as an adaptive sampling rate mechanism as well a distributed architecture with multiple collectors. We also plan to evaluate EMC2 for its network coverage when it employs its adaptive sampling rate mechanism.

REFERENCES

[1] N. McKeon *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," in *ACM SIGCOMM CCR*, April 2008.

[2] M. A. Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *USENIX NSDI*, 2010.

[3] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers," in *IFIP Networking*, 2011.

[4] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware steady state VM management for data centers," in *IFIP Networking*, 2012.

[5] "Remote monitoring(rmon)." [Online]. Available: http://www.cisco.com/en/US/tech/tk648/tk362/tk560/tsd_technology_support_sub-protocol_home.html

[6] "sflow." [Online]. Available: http://www.sflow.org

[7] "Netflow." [Online]. Available: www.cisco.com/go/netflow

[8] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," in *SIGCOMM*, 2004.

[9] B.-Y. Choi and S. Bhattacharyya, "Observations on cisco sampled netflow," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 3, Dec. 2005.

[10] L. Deri, E. Chou, Z. Cherian, K. Karmarkar, and M. Patterson, "Increasing data center network visibility with cisco netflow-lite," in *Proceedings of the 7th International Conference on Network and Services Management*, ser. CNSM '11, 2011.

[11] V. Sekar, M. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. Andersen, "CSAMP: A System for Network-Wide Flow Monitoring," in *USENIX NSDI*, April 2008.

[12] V. Sekar, M. Reiter, and H. Zhang, "Revisiting the Case for a Minimalist Approach for Network Flow Monitoring," in *ACM IMC*, November 2010.

[13] T. Benson, A. Anand, A. Akkela, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *ACM CoNEXT*, December 2011.

[14] A. Curtis *et al.*, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *IEEE INFOCOM*, 2011.

[15] "Netflow support in vsphere 5.0." [Online]. Available: http://www.vmware.com/files/pdf/techpaper/Whats-New-VMware-vSphere-50-Networking-Technical-Whitepaper.pdf

[16] "Open vSwitch - An Open Virtual Switch," http://www.openvswitch.org.

[17] "Kvm." [Online]. Available: www.linux-kvm.org/page/Main_page

[18] "Xen." [Online]. Available: www.xen.org

[19] "Microsoft hyper-v whitepaper." [Online]. Available: http://download.microsoft.com/download/5/D/B/5DB1C7BF-6286-4431-A244-438D4605DB1D/WS%202012%20White%20Paper_Hyper-V.pdf

[20] "Random sampled netflow." [Online]. Available: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/nfstatsa.html

[21] "Ibm systems networking rackswitch g8264 application guide." [Online]. Available: http://www.bladenetwork.net/userfiles/file/G8264_AG_6-6.pdf

[22] "Differentiated services code point (dscp) - rfc 2474." [Online]. Available: http://www.ietf.org/rfc/rfc2474.txt

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

[24] "nmon for linux." [Online]. Available: http://nmon.sourceforge.net/pmwiki.php

[25] "Iperf Bandwidth Measurement Tool," http://sourceforge.net/projects/iperf/.

[26] "Recommended sampling rates." [Online]. Available: http://blog.sflow.com/2009/06/sampling-rates.html