On Time-Stamp Accuracy of Passive Monitoring in a Container Execution Environment

Farnaz Moradi, Christofer Flinta, Andreas Johnsson, Catalin Meirosu
Cloud Technologies, Ericsson Research, Sweden
Email: {farnaz.moradi,christofer.flinta,andreas.a.johnsson,catalin.meirosu}@ericsson.com

***Abstract*—Passive monitoring of network performance parameters is experiencing a revival due to widespread adoption of virtualization and software-based implementation of network functions. Timestamping is one of the most challenging operations needed for passively monitoring network traffic performance parameters such as latency and jitter. We develop a setup whereby functions that monitor the network traffic are deployed in monitoring containers adjacently to, and interconnected through a virtual switch with the monitored Virtual Network Function instance. In this scenario, we evaluate the effects of container virtualization and virtual switch mirroring of traffic on the measurement of latency. The evaluation results indicate very low measurement errors (a few microseconds in our testbed) which are consistent over different measurement scenarios, thus validating the feasibility of this technique for passively monitoring latency.**

I. Introduction

Advances in virtualization have led to the emergence of network function virtualization (NFV) which decouples network functions from dedicated hardware to software-based applications that can run on commercial off-the-shelf hardware. Accurate, timely, and non-intrusive monitoring of network performance metrics of such virtual network functions (VNFs) is critical for identifying and avoiding violation of performance guarantees. In recent years, container-based virtualization has received considerable attention and has become a candidate for running VNFs. Container execution environments provide lightweight virtualization with much lower overhead compared to hypervisor-based virtualization with Virtual Machines (VMs), since rather than running a full operating system containers share the same kernel with the operating system of the host machine [19], [21]. Moreover, containers are created and removed much faster compared to VMs, and container-based virtualization provides better resource utilization since idle containers do not use any resources.

In Linux containers, the kernel-level namespaces are used for resource isolation and the Control groups (cgroups) [13] are used for managing and limiting resources. Cgroups also expose resource usage metrics such as memory, CPU, and block I/O which can be used for monitoring purposes. Such metrics for the containers running in a host machine can be collected by a separate container in the host, e.g., cAdvisor[1].

Existing tools for monitoring containers only provide compute resource utilization statistics as well as counters for packets and bytes on the interfaces of the containers. Network-related metrics, such as per-flow metrics can be measured by for example enabling sFlow [16] on the virtual switch infrastructure that interconnects the containers. SFlow is a general purpose monitoring tool for sampling packets and interface counters on packet forwarding devices. However, sFlow does not provide end-to-end measurements of metrics such as delay, jitter, and packet loss [2].

End-to-end measurements of network performance can be performed using active or passive measurement methods. In active measurements, test packets are injected from a probe in the network and are received in another probe. In passive measurements, in contrast, actual network traffic is being observed without injecting probe packets. In addition to network performance monitoring, passive measurement methods are also used for example to perform traffic analysis for traffic profiling, classification, and characterization, anomaly and intrusion detection, and debugging.

One option for measuring end-to-end network performance metrics in containers is to run monitoring functions inside the same container in which the VNF application is being executed. This requires the monitoring code to be added to the image of the VNF or executed inside the container after instantiation of the VNF container. Another option, which we investigate in this paper, is to execute monitoring functions in separate and adjacent monitoring containers. The monitoring container receives a copy of packets originated from or destined to the VNF instance and calculates different network performance metrics.

Running the monitoring functions in a separate container instead of running them inside the VNF container has many advantages. Some of these advantages are as follows:

1) The monitoring can be done transparently without a need to run any process inside the VNF container or instrument the VNF image with required software.
2) A single monitoring container can be used to monitor multiple VNF containers in contrast to running a monitoring process in each VNF container.
3) Using a monitoring container enables separation of the monitoring process from the VNF processes by assigning them to different CPU cores.
4) Monitoring is isolated from VNF so failure of the monitoring process will not adversely affect the VNF container.
5) A separate container for monitoring allows more flexibility in running different monitoring functions so that they can be updated, re-configured, or changed

[1] https://github.com/google/cadvisor

ISBN 978-3-901882-83-8

on-the-fly transparently to the VNFs.
6) Migration of the VNF instance can be done independently and without affecting the monitoring.
7) The monitoring container can be controlled and managed by the infrastructure provider who may be different from the VNF provider.
8) The monitoring container can create log files from the VNF traffic which can be used for debugging the application instead of the application itself generating the logs.

Despite many advantages of running monitoring functions in separate monitoring containers, the effect of such setup on the accuracy of timestamps required for passive network monitoring is not known. Previous studies have shown that hypervisor-based virtualization affects the timestamping and performance measurement accuracy [5] [14]. However, none of the previous studies have looked into the effects of container-based virtualization on the accuracy of measurements, particularly the accuracy of the timestamps required for passive network performance monitoring.

In this paper, our contributions are as follows. We develop a setup for passive monitoring of network traffic by introducing standalone monitoring containers which are interconnected through a virtual switch with the monitored VNF containers. We study the effect of container virtualization and packet copying in virtual switches on the accuracy of timestamps obtained by the monitoring functions running inside the monitoring containers. Moreover, we evaluate the accuracy of passive latency monitoring using different methods in our setup. Our measurement results indicate low and consistent timestamping errors over different measurement scenarios, therefore confirming that our monitoring setup is suitable for passive monitoring in container execution environments.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes our measurement system for passive container-based monitoring and Section IV presents our testbed and experimental settings. In Section V the evaluation method is described and the experimental results are presented. Section VI presents a discussion of our findings. Finally, Section VII concludes the paper.

II. Related Work

Existing tools for monitoring containers only gather metrics such as CPU, memory, and block I/O usage for containers running in a host machine. These metrics can be obtained from cgroups in Linux [13]. Network metrics which can be collected by existing tools are limited to the number of packets and bytes received/transmitted from an interface. Although sFlow [16] can be used for sampling packets and interface counters, it does not provide network metrics such as latency.

Active and passive measurements of network performance metrics can be performed using OpenFlow messages in OpenFlow-enabled virtual and physical switches in the network. Examples of such methods are latency measurements [17] and link utilization monitoring [23], however these methods are not in the scope of this paper since they are not designed for passive measurements for traffic analysis.

The effects of virtualization on network measurements have been previously studied. In [14] the authors studied how timestamping variability is affected by hypervisors and showed that only under certain conditions timestamping performance in virtualized environments gives good results. In [5] the effects of Kernel-based Virtual Machine (KVM) virtualization on active round trip time measurements have been evaluated. The results show that the measurements are affected by both CPU load in the host and load in the network. Although different studies have investigated the effects of virtualization on timestamp accuracy of measurements, to the best of our knowledge effects of container-based virtualization on passive measurements have not been investigated before.

A passive network monitoring function which requires accurate timestamping is latency. In the rest of this section we mention some related work for passive latency monitoring. Note that none of these methods have been designed or evaluated in virtual platforms such as VMs and containers.

A naïve way to passively measure latency is to store packet hashes together with timestamps at both the sender and receiver side and periodically exchange and compare the hash and timestamp values. However, such an approach enforces high storage and communication overhead. In order to overcome the problems of the naïve approach, a variety of efficient passive latency measurement methods have been proposed in the literature. These methods can be roughly divided into three categories: aggregate, per-flow, and per-packet methods.

In [9] an aggregate method has been proposed, where a Lossy Difference Aggregator (LDA) data structure is created at both the sender and the receiver side and at the end of each measurement interval is exchanged for calculating loss and latency. LDA requires synchronization packets to be sent over the same channel as the data traffic and makes an assumption that the packets arrive in FIFO order. FineComb [12] also calculates aggregate latency but rather than making an assumption about packet ordering proposes a stash data structure to recover from packet reordering.

Aggregate latency measurements cannot capture the latency experienced by different flows, therefore more fine-grained latency measurement methods have been proposed in the literature. Reference Latency Interpolation (RLI) [10] obtains per-flow latency measurements, however it requires injecting reference packets (probe packets). MAPLE [11] presents an architecture for latency monitoring where the granularity of the measurements can be selected. In MAPLE the focus is on delay storage and query and it is assumed that the packets can carry timestamps. In [18] COLATE, a counter-based per-flow latency estimation scheme for latency monitoring without using any probes or timestamps inside packets has been proposed. COLATE can achieve accurate results with low overhead compared to previously existing methods. More details about this method can be found in Section V-B3.

In addition to per-flow measurements more fine-grained results can be achieved using per-packet methods in exchange for extra costs. For example, in [22], an approach for per-packet delay and loss measurements has been proposed which first uses an order preserving aggregator (OPA) data structure to transmit ordering information for recovering from packet reordering and then sends the compressed packet timestamps to be used for estimating delay.

In this paper, rather than proposing a new latency monitor-

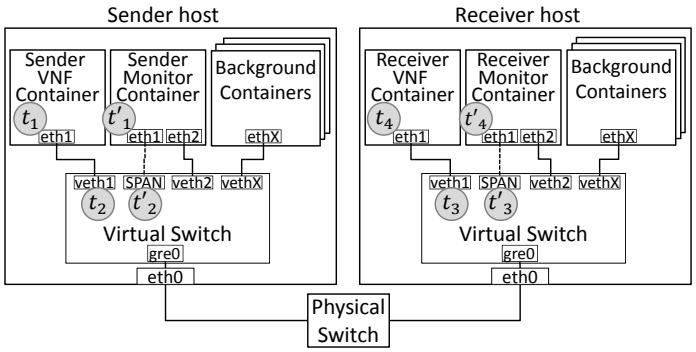


Fig. 1: Experimental setup with two host machines with Open vSwitch and Docker containers.

ing algorithm, we investigate how existing methods perform in a container execution environment. As part of the evaluations presented in our paper, we use the data structures proposed by COLATE to implement a passive latency monitoring tool running inside monitoring containers adjacently to the monitored VNF container instances.

III. Measurement System and Challenges

In this section, we present our setup for passive monitoring of VNFs and applications running in a container execution environment. In this setup, a monitoring container is instantiated and attached to the same virtual switch to which the VNF is connected. The virtual switch is then configured to duplicate the packets originated within or destined to the VNF container and send the copies to the monitoring container.

In general, duplicating and copying packets for passive network monitoring can be performed in two different ways: in-line mode and mirroring mode [7]. In the in-line mode, a network tap (a.k.a. Test Access Port (TAP)) is used to duplicate all traffic passing through it and provide a connection to the capturing device. In mirroring mode, a network switch duplicates the packets from one or more ports and sends the replicates to a single monitoring port (a.k.a. Switched Port analyzer (SPAN) port) to be captured for analysis.

The monitoring function which is executed inside the monitor container receives the copy of VNF packets on its interface. The monitoring function can then perform filtering and sampling of packets before using them for estimating different network performance metrics. In our setup, the monitoring functions in different hosts can also communicate with each other via a separate interface in order to exchange control and synchronization messages which are required for different types of measurements such as packet loss and latency.

The copying of packets can affect the accuracy of the packet timestamps. Our main focus in this paper is to investigate the accuracy of packet timestamps observed at the monitoring containers in comparison to what is observed inside the VNF containers. Therefore, we calculate timestamping error as the difference of a packet timestamp observed in the VNF and monitor containers. Figure 2 shows how the timestamping errors on the sender and the receiver hosts are calculated. Further, the effect of loading different resources on the host machines on the accuracy of the timestamps observed in the monitoring containers is investigated.

Additionally, we evaluate the effect of timestamp accuracy on latency measurements between different hosts. Latency can be passively measured by either round trip time (RTT) or one-way measurements. In passive RTT measurements, the request and reply packets of a flow are matched with each other and their timestamps are compared in order to estimate the RTT. However, passive RTT measurement is not always possible, e.g., for UDP communications. Passive one-way latency measurements can be used for any type of traffic where the timestamps of packets at the sender and the receiver sides are compared against each other.

A common assumption made by passive one-way latency measurement methods, is that the clocks at the sender and the receiver hosts are tightly synchronized, since unsynchronized clocks lead to errors in the measured values. In this paper we study the effect of running passive latency functions in separate containers and compare the results with the latency values which could be obtained from inside the VNF containers as shown in Figure 2. Errors caused by synchronization problems are not part of this study. More information about general synchronization issues and achieving high precision can be found in [3] and [20], respectively.

IV. Testbed setup

Figure 1 shows the testbed used for evaluating the container-based passive monitoring setup presented in this paper. This testbed is comprised of two physical host machines that are connected with a physical switch. Each host runs a VNF container and a monitoring container as well as a number of background containers. The virtual switches on the host machines are connected to each other with a Generic Routing Encapsulation (GRE) tunnel and each switch is responsible for tapping or mirroring the VNF packets to the SPAN port, so that a monitoring function which runs in the monitoring container can capture and analyze the packets. The background containers are used for testing purposes such as generating background CPU and network load.

In our measurements, we have used Docker containers[2] and Open vSwitch (OVS)[3] virtual switches. When a Docker container is created, a pair of peer interfaces is created where one peer becomes the interface for the container and the other one is bound to the virtual switch. OVS is an OpenFlow switch which uses flow classification and caching techniques to provide high performance forwarding [15].

In OVS, the data plane is implemented in kernel space while the control plane is in user space. OVS manages packets as flows where the first packet of a flow is sent to the controller. The controller determines how the packet should be handled and passes the packet back to the data plane. Additionally, the controller updates the cache in the kernel module so that the rest of the packets of the flow can only go through the data plane following the given instruction. This means that the first packet of a flow which has to cross the boundary of kernel and user space and be classified by the controller

[2]https://www.docker.com/
[3]http://openvswitch.org/

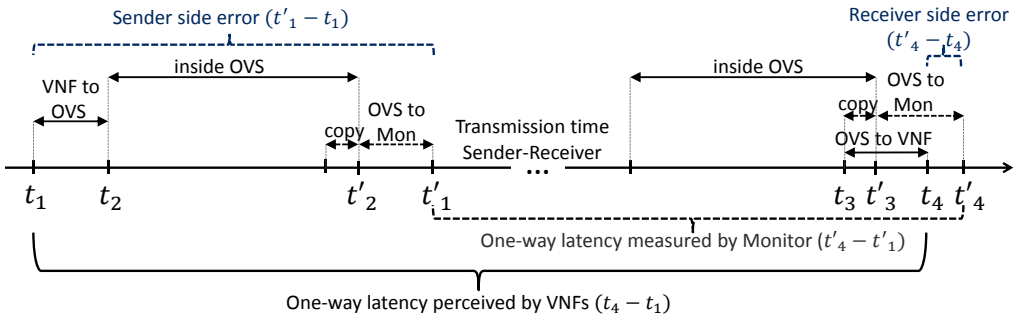


Fig. 2: Timestamp errors on sender and receiver sides.

experiences longer processing time in the switch compared to the rest of the packets. The flows in the kernel cache that have been idle for a configured amount of time are then removed by the controller [15].

Using OVS allows us to perform both tapping and mirroring of traffic. Tapping can be done by adding a flow to the OpenFlow switch to perform two actions on each packet coming from the VNF port. One action is to forward the packet normally, and the second action is to send it to the monitoring port (e.g., *ovs-ofctl in_port=1,actions=NORMAL,output:2*). Mirroring is supported by OVS which is implemented in user space through modification of the same flow table exposed through OpenFlow and can be used by adding a mirror to the virtual switch and defining the input and output ports.

In order to evaluate the effects of container-based virtualization and packet duplication on timestamp accuracies, we have used the *tcpdump* tool for obtaining packet timestamps on different capture points. Figure 1 shows the capture points, i.e., interfaces on which we used *tcpdump*. On the sender side, *tcpdump* is executed inside the VNF container to obtain timestamp t_1, on the host machine to obtain timestamps t_2 and t'_2, and inside the Monitor container to obtain t'_1. The timestamps on the receiver side are obtained similarly, i.e., t_3 and t'_3 on the host machine and t_4 and t'_4 in the VNF container and the monitor container, respectively.

In our experiments, different tools have been used to study the effects of loading resources on the measurements. The CPU, I/O, and virtual memory loads are generated using the *stress* tool [1], where we start multiple background containers running the tool. The network load on the virtual switches has been emulated by using the *tcpreplay* tool and replaying random pcap files from inside the background containers. We have also used *tcpreplay* to cause network congestion on the path between the host machines.

V. Evaluation

In this section we present our evaluation method and the experimental results. The measurements are divided into two subsections: (V-A) Timestamping errors, and (V-B) Passive latency measurements. In subsection V-A we present measurement results for three scenarios: (1) measurements with no background load, (2) measurements with loading different resources such as CPU, I/O, and virtual memory, and (3) measurements with network load on the virtual switch. In subsection V-B results for (1) passive round trip time measurements, (2) passive one-way measurements, and (3) passive one-way measurements using COLATE [18] are presented.

A. Timestamping errors

In this section we present our measurement results on comparing the timestamps of packets sent by a VNF container with the timestamps of the copy of the packets captured in the corresponding monitor container. The goal is to investigate the effects of container-based virtualization as well as switch processing and packet copying on the accuracy of the timestamps.

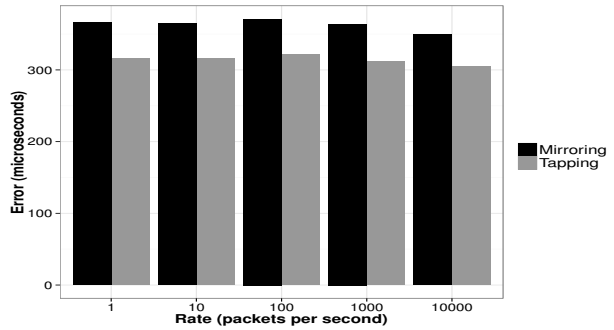
For each measurement, we emulate the VNF traffic by running the ICMP *ping* tool with different frequencies inside the sender VNF container which allows us to also measure RTT between the VNFs. The *tcpdump* tool is used both in the VNF and in the monitor container to capture the echo request messages. In our experiments, the main focus is on measuring the errors caused by container and switch virtualization, which are not dependent on the type of VNF traffic. For each packet, we calculate and report the timestamping errors. During our measurements we observed that when ping is running with higher packet send rates, it returns lower RTT values regardless of where it is running, i.e., inside the host or inside a container. For example, we observed that pinging the receiver VNF container from inside the sender VNF container with frequency of one packet per second with packet size of 64 bytes returns an average RTT of 485 μs compared to 331 μs for 1000 packet per second rate. This is in line with results observed in [5].

1) Measurements with no load:

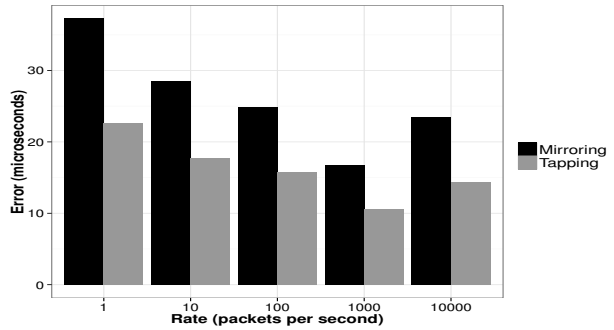
The first set of experiments is performed with no background container and no extra network load. The main goal of these measurements is to study the errors caused by the container virtualization and virtual switch processing including packet mirroring/tapping.

Figure 3 shows the timestamping errors on the sender host comparing tapping and mirroring. The y-axis shows the measurement error, where for each packet we calculated the error as the difference between the timestamps reported by *tcpdump* in the VNF and in the monitor, i.e., $t'_1 - t_1$ as shown in Figure 2. The x-axis shows the send rates given as input to the ping tool running inside the VNF container. We have observed that the ping does not send the exact number of packets with the input send rate, e.g., for input send rates 100 and 10000 packets per second, ping sent around 84 and 27000 packets per second, respectively. However, in the figures the input parameter for ping is shown, indicating the order of magnitude for the send rates.

It can be seen that the time it takes for the first packet (Figure 3a) to be received by the monitoring container is

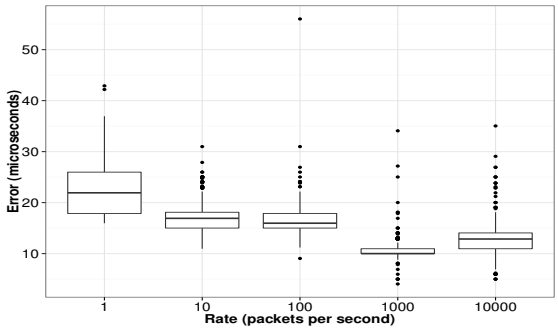


(a) First flow packet

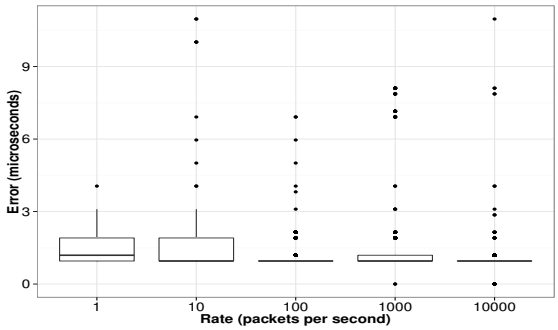


(b) Mean for the rest of the flow packets

Fig. 3: Comparison of the error caused by tapping versus mirroring packets to the monitoring container on the sender side for different send rates with no background load ($t'_1 - t_1$).



(a) Sender host ($t'_1 - t_1$)



(b) Receiver host ($t'_4 - t_4$)

Fig. 4: Comparison of the error on sender and receiver side for different send rates (excluding the first flow packet).

significantly higher than the average value for the rest of the packets (Figure 3b). On average the error was 328 μs and 375 μs for the first packet using tapping and mirroring respectively, with standard deviation of 28 μs and 25 μs. For the rest of the packets in each flow the error was on average 16 μs and 26 μs with standard deviation of 2 μs and 4 μs using tapping and mirroring respectively. As mentioned in Section IV, for each new flow OVS sends the first packet to the controller in the user space and the rest of the packets are forwarded in the data plane in the kernel space. Moreover, it can be seen that the timestamps obtained from tapping of packets are slightly closer to the timestamps perceived by the VNF container compared to mirroring, i.e., smaller error values. In the rest of our measurements we only use tapping because of the lower error rate. The figure also shows that the error for lower send rates, particularly one packet per second, is higher than the measurements with higher send rates. As mentioned above, ping returns higher RTT values for low send rates which also affects the measurement results and is the reason for higher errors shown in the figure.

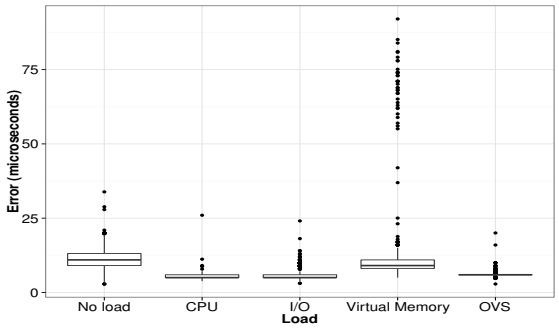
Figure 4 shows the timestamping error on the sender and the receiver hosts for different send rates. It can be seen that the errors on the sender side are much higher than the errors on the receiver side. The main reason for the differences, as shown in Figure 2, is that on the sender side each packet is first observed in the VNF (t_1) and then enters the switch (t_2) where it is processed and then is duplicated and its copy is sent to the monitor interface (t'_2) and is observed by the monitoring function (t'_1), while on the receiver side the switch processing time affects each packet and its copy similarly. Additionally, in the receiver the time it takes for the packets to be observed inside the VNF (from t_3 to t_4) and the receiver monitor container (from t'_3 to t'_4) are equal, and the main factor that causes errors is the packet duplication time in the switch which is required for both tapping and mirroring.

In our measurements, we observed that running *tcpdump* in all the four capture points inside the sender host increases the errors compared to the experiments where *tcpdump* was only executed inside the containers. For example, we observed that when *tcpdump* was capturing traffic on all the measurement points on the sender side, the total timestamping error was around 14 μs higher than the measurements where *tcpdump* was only running inside the containers. This observation shows that *tcpdump* also affects the accuracy of our measurements.

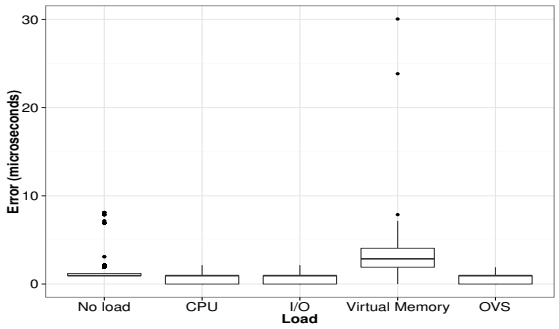
In order to estimate the time it takes for packets to reach the OVS interface from the interface inside the container and vice versa without running *tcpdump* in all four capture points, we performed the following measurements. We calculated the values for $t'_2 - t_2$ when *tcpdump* was only running on the host as well as the values for $t'_1 - t_1$ when *tcpdump* was only running inside the containers. The values obtained from the capture points in the host were subtracted from the values obtained from the containers to estimate the time $(t_2 - t_1) + (t'_1 - t'_2)$. The results indicate that the time it takes for packets to arrive from the VNF container to the OVS and from the OVS to the monitor container is less than 0.8 μs in average with standard deviation 0.5.

2) Effect of Resource load:

In this section we study the effect of loading physical



(a) Sender host ($t'_1 - t_1$)



(b) Receiver host ($t'_4 - t_4$)

Fig. 5: The effect of high loads on host resources on per-packet timestamping errors excluding the first packet.

resources on the host machines on the accuracy of the packet timestamps obtained by the monitoring containers. In order to load the resources on the host machine, we started eight background containers running the *stress* tool to load different resources. We used the *stress* tool to spawn a given number of workers spinning on *sqrt()* to load the CPU, *sync()* to load I/O, and *malloc()*/*free()* to load the virtual memory.

Figure 5 shows the results of loading CPU, I/O, and virtual memory. It can be seen that loading memory increases the error caused by copying packets on both the sender and the receiver sides, while loading CPU or I/O in average reduces the time compared to measurements with no background load.

The main reason for reduced error when physical resources including CPU are loaded is due to CPU auto scaling which happens when the system is under load. By default the CPU governor is set to "ondemand" which allows CPU to achieve maximum clock frequency when the load is high and achieve minimum frequency when the system is idle which allows the system to adjust power consumption according to system load.

3) Effect of network load in virtual switch:

In this section we study the effect of network load in the virtual switch on the accuracy of the packet timestamps. We used the *tcpreplay* tool to load the virtual switch with network traffic. In the measurements, eight background containers were attached to the OVS and replayed traffic from a given pcap file with maximum possible speed. The traffic was forwarded to a 9th background container by the OVS in the host where the packets were received and dropped. During these experiments, the CPU on the host machine also reached maximum load but we did not observe any ICMP packet loss.

Figure 5 shows the results for this scenario with network load in the virtual switch (OVS). It can be seen that the error caused by copying packets is very similar to the experiments with CPU loading.

B. Passive latency measurements

Latency monitoring requires accurate timestamping. In this section we evaluate the effect of our passive container-based monitoring setup on the accuracy of round trip time (RTT) measurements and one-way latency measurements.

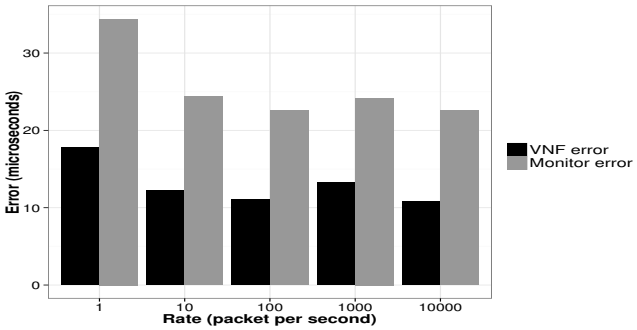
For RTT measurements, we use the values obtained from ICMP pings, which were sent from inside the sender VNF container to the receiver VNF container, as the ground truth since it is the actual RTT experienced by the VNFs. Then we evaluate the measurement results from both inside the VNF container and the monitor container on the sender side. For passive latency measurements, we use the latency values perceived by VNF containers as the ground truth and evaluate the accuracy of the latency values reported by the monitoring containers. Even though the ground truth one-way latency values are affected by the precision of time synchronization, the synchronization affects the latency calculated by the monitor containers similarly, and therefore it does not affect the reported error values. Moreover, we compare the values obtained from the above naïve approach with the latency values measured by a state-of-the-art passive monitoring algorithm which is implemented by using the data structures and the average latency estimation method presented in [18].

The measurements are done for two scenarios: with no background traffic and with network load which causes congestion on the path between the host machines. In these measurements a background container in the sender host sends packets with a high rate and the traffic is received and dropped by the receiver host. The high send rate causes network congestion and leads to VNF packet loss. However, the loss rates did not change with different VNF send rates.

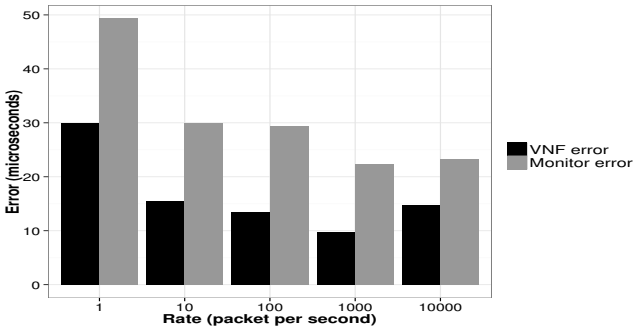
1) Passive Round Trip Time:

The RTT for each packet as perceived by the VNF containers is calculated by comparing the timestamps of each *echo request* and *echo reply* packets with the same sequence number, (i.e., $RTT_{VNF} = t_1(reply) - t_1(request)$) where $t_1(request)$ is the time when an echo request packet was sent from the sender VNF container and $t_1(reply)$ is the time that the echo reply packet corresponding to that request is received by the sender VNF container. Similarly the RTT values as perceived by the sender monitor container are calculated as $RTT_{Mon} = t'_1(reply) - t'_1(request)$. In our measurements, the average RTT reported by ping for different send rates was 359.8 μs compared to 346.8 μs and 334.2 μs as reported by the VNF and the monitor containers, respectively.

Figure 6 shows the measurement errors of the RTT measurements for VNF and monitor container versus the ground truth, i.e., values returned by ping. It can be seen that even the RTT values reported by the VNF container using *tcpdump* are underestimated compared to the ground truth. The reason for underestimation is the delay for the request packet to be sent from the ping application to the interface where *tcpdump* records a timestamp, as well as the delay for the arriving reply

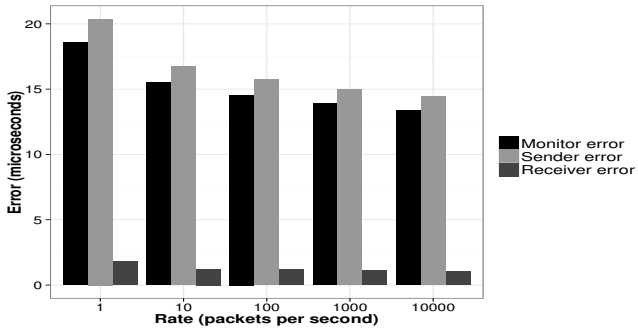


(a) No load

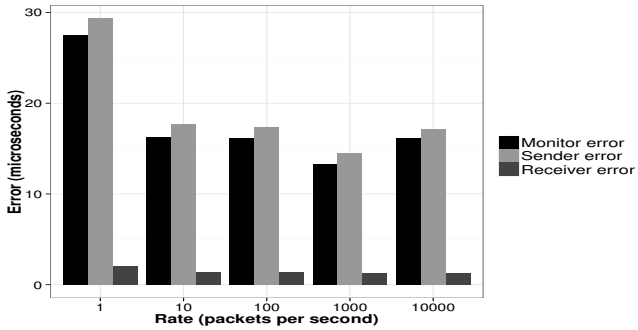


(b) Network Congestion

Fig. 6: Error of RTT measurements calculated from inside the VNF and monitor containers compared to the ground truth values reported by ping, i.e., VNF error ($RTT_{ping} - RTT_{VNF}$) and monitor error ($RTT_{ping} - RTT_{Mon}$).



(a) No load



(b) Network Congestion

Fig. 7: Error of latency measurements calculated from inside the VNF containers and from inside the monitor containers ($L_{VNF} - L_{Mon}$) in comparison with the timestamping errors on the sender side ($t'_1 - t_1$) and the receiver side ($t'_4 - t_4$).

that is first timestamped by *tcpdump* and then by the ping tool. It should be noted that the errors of the values reported by the monitoring container compared to the ground truth also include the VNF errors. Overall, it can be seen that the errors of the measurements in the monitoring container are consistent for different measurements. Moreover, even though in experiments with network congestion 63% of packets were lost and the RTT reported by ping was increased to 28.5 ms, the error values were only slightly increased.

2) Passive latency:

The one-way latency for each packet as perceived by the VNF containers is calculated as $L_{VNF} = t_4 - t_1$ and is used as the ground truth. The latency values as perceived by the monitor containers are calculated as $L_{Mon} = t'_4 - t'_1$. The actual values reported by one-way latency measurements are not accurate due to drifting clocks (we have used NTP for time synchronization which does not provide the required microsecond synchronization precision for our measurements). Nevertheless, the obtained passive one-way measurement values can be used for identifying latency changes over time and detecting congestion in the network.

Figure 7 shows the error values for one-way latency measurements. Our measurements show that the latency values calculated from inside the VNF containers and from inside the monitor containers are very close to each other and the measured latency is only slightly underestimated, in average around 15 μs. These results are in line with what was observed for RTT measurements, where the absolute average error for values reported from the monitor and the VNF was around 13 μs. The figure also shows the sender and the receiver side timestamping errors. It can be seen that the errors on the sender side are much higher than the errors on the receiver side. As depicted in Figure 2, the total error of monitored one-way latency values are caused by the error which happens due to packets arriving in the sender monitor after the copying and processing times on the sender side and the error which happens due to packets arriving in the receiver monitor after being copied in the switch.

In the measurements with network overload, the heavy load on the link caused in average 62% packet loss for the VNF traffic in our measurements. The loss rates were caused by the background traffic and did not change for different VNF send rates. Figure 7 shows that the increase in network load causes the average per packet latency error to only slightly increase compared to the tests with no background network load.

3) Passive latency measurements with COLATE:

In the measurements presented in the previous sections, every single packet has been compared on the sender and receiver side. However, such naïve approach is not suitable for automatic and continuous passive latency measurements due to high storage and communication cost.

COLATE [18] is a counter-based per-flow latency estimation scheme which "allows noise in recording times for minimizing storage space, and then statically de-noises the recorded information for obtaining accurate latency estimates". The COLATE scheme consists of two phases: recording and

querying. In the recording phase, for each packet at the measurement point, a timestamp is recorded using a vector called counter vector. Moreover, each unique flow is mapped to a unique subset of these counters called counter subvectors. In other words, each packet is first mapped randomly to a counter in the counter subvector of its corresponding flow, and then the current time is added to that counter.

The average latency for flow f in COLATE is estimated as $\tilde{\mu}_f = 1/p_f(\tilde{t}_{f,R} - \tilde{t}_{f,S})$, where $\tilde{t}_{f,X} = 1/(n-m)\{n\Sigma_{j=1}^{m} S_X^f[j] - mt_X\}$ is the estimated sum of all timestamps of the packets in flow f which is calculated both at the sender ($\tilde{t}_{f,S}$) and the receiver ($\tilde{t}_{f,R}$). In these equations, S_X^f denotes the counter subvector of flow f, n denotes the number of counters in the counter vector C_X, m denotes the number of counters in S_X^f, t_X denotes the sum of all counters in C_X, and p_f denotes the number of packets in flow f.

In [18], it is shown that COLATE achieves high accuracy compared to other existing methods using simulations and real network traffic. In this study we use COLATE as an example of an accurate and low-overhead passive latency measurement method and investigate the effect of our monitoring setup on its accuracy.

We have implemented the COLATE method to be executed in measurement sessions instead of the original design with two separate recording and querying phases. In the start of each session, the sender monitor function sends a start message to the receiver and starts collecting packets using *libpcap* library. The receiver side similarly starts passive data collection after receiving the start message. At the end of the interval, the sender sends a stop message, together with the estimated sum of all timestamps $\tilde{t}_{f,S}$. Upon arrival of the stop message, the receiver compares its estimated sum of all timestamps $\tilde{t}_{f,R}$ with the value received in the message. The difference gives us the estimated average latency over the measurement session. In our experiments we only run tests for a single flow measurement in the container-based setup. In the original design of COLATE, the number of packets per flow p_f is obtained from a separate tool. In our implementation this value is also calculated by the monitor function itself. Moreover, in our implementation the stop message also carries an incremental stream digest value, i.e., a hash value created from all the packets in the measurement session, which allows us to make sure that the same set of packets are being compared in each session.

Figure 8 shows the one-way latency errors for different measurements with different session lengths. The x-axis shows the number of packets that were collected in each session. The bars in the figure show the errors between the average latency value calculated by COLATE and the average latency from per-packet measurements from inside the monitor container ($L_{COL} - L_{Mon}$). It can be seen that the error of measurements performed by COLATE depends on the number of packets in each measurement session. Our measurements show that COLATE slightly overestimates the latency values compared to per-packet values obtained from the monitoring container. Overall, one should take into consideration that the total error of measurements using this algorithm is the sum of the overestimation of COLATE and underestimation caused by using our monitoring setup with adjacent monitoring containers.

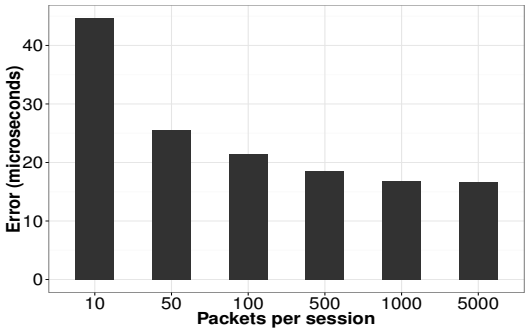


Fig. 8: Error values for average latency by using COLATE compared to a naïve method with different session lengths.

VI. Discussion

The monitoring setup presented in this paper can be used for different types of measurements. Some measurements such as packet loss detection do not require accurate timestamping. Other types of measurements such as available capacity measurements [4] require calculating the time difference between packets. Further, passive latency calculation is a challenging problem which also requires accurate timestamping. In this study we have evaluated a container-based measurement setup and how it affects the accuracy of passive latency estimation.

In this paper, we have performed different measurement experiments to evaluate the accuracy of measurements in a setup corresponding to real-world deployment of VNFs. In the measurements, ICMP ping was used to emulate VNF traffic. We also observed similar measurement results when the VNF containers where sending UDP traffic. This observation confirms that the type of traffic being passively monitored does not affect the accuracy of the timestamps in our monitoring setup. A future study could be performed with different packet sizes and random send intervals to get a more detailed view on the impact from packet size and send patterns on the timestamping accuracy. We have also shown that the main cause of error in timestamping information is due to switch packet processing on the sender side. It is expected that by continuous enhancement and optimization of virtual switch implementations, these errors caused by switch processing will be reduced. Our measurements have also shown that under heavy CPU loads, the timestamping errors caused by both container and switch virtualization are reduced. The load on the CPU has also caused the ping tool to report lower RTT values compared to experiments with no background load. These observations are due to CPU auto scaling which increases the clock frequency of CPU on the host machine. This means that by increasing the clock frequency of CPU, it is possible to reduce the timestamping errors and achieve more accurate results in expense of losing power saving benefits. Additionally, we observed that pinning the monitoring container to a CPU core can reduce the measurement errors, particularly in measurement scenarios where virtual memory is loaded.

In previous studies it has been shown that port mirroring has a number of drawbacks such as adding additional burden on the CPU of the switch [8]. In our experiments the extra CPU burden added by packet duplication was negligible. Moreover, it was shown in [24] that mirroring may introduce delay, loss,

and reordering of packets due to buffering of packets before forwarding them. In our measurements we did not observe any packet loss due to mirroring or tapping. However, we observed re-ordering of the packets received by the monitoring container compared to the packets sent from a VNF container both for the tapping and mirroring experiments. The re-ordering happens among the packets that belong to different flows and not the packets that belong to the same flow. Therefore, if the monitoring function or the traffic analysis tool requires accurate ordering of the packets for different flows, using mirroring/tapping in OVS can lead to inaccurate results.

An interesting future direction would be to evaluate the performance of different types of virtual switches. Initial experiments comparing OVS with original Docker bridge has shown that although the time it takes for the first packet to be processed is higher than the rest of the packets in both of the switches, the processing time for OVS is much higher. Overall, the average processing time it takes for all packets of a flow to be processed by the Docker bridge is lower that the OVS. Additionally, Docker allows a container to share the network stack of another container instead of directly connecting to the bridge. This means that a monitoring container can be connected to the network stack of a VNF container and perform passive measurements. The advantage of such implementation is that the errors caused by processing and copying packets by virtual switch are eliminated. However, this approach to implementation has many disadvantages including violating isolation and separation of containers.

Overall, the timestamping error caused by our setup is very low compared to the latency values reported in real datacenters. The latency values reported in [6] show that in a normal working day without any network incidents, the 50th percentile intra-pod and inter-pod round-trip latencies (from physical servers and not through virtualization layers) are around 216 μs and 268 μs, respectively. These observations also confirm the feasibility of using our setup for passive latency monitoring.

VII. Conclusions

In this paper we presented and evaluated a container-based monitoring setup for passive monitoring of VNF traffic. Our main focus has been on the timestamping accuracy which is required for passive monitoring of different network performance metrics such as latency. The key finding of our study is that the main source of error for passive container-based monitoring is caused by the switch processing time on the sender side. Moreover, we observed that the errors are quite stable and are not affected much by the load on the host machines and by congestion on the network. The results indicate that by doing initial tests and obtaining error estimates, one can calibrate a monitoring system.

References

[1] Stress tool, [online]http://people.seas.harvard.edu/~apw/stress/.

[2] Internet protocol data communication service-ip packet transfer and availability performance parameters, itu-t recommendation Y.1540, 2011.

[3] T. Broomhead, L. Cremean, J. Ridoux, and D. Veitch. Virtualize everything but time. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 1–6, 2010.

[4] S. S. Chaudhari and R. C. Biradar. Survey of bandwidth estimation techniques in communication networks. *Wirel. Pers. Commun.*, 83(2):1425–1476, July 2015.

[5] R. Dantas, D. Sadok, C. Flinta, and A. Johnsson. Kvm virtualization impact on active round-trip time measurements. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 810–813, May 2015.

[6] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *ACM SIGCOMM Conference*, pages 139–152. ACM, 2015.

[7] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *Communications Surveys Tutorials, IEEE*, 16(4):2037–2064, Fourthquarter 2014.

[8] W. John, S. Tafvelin, and T. Olovsson. Review: Passive internet measurement: Overview and guidelines based on experiences. *Comput. Commun.*, 33(5):533–550, Mar. 2010.

[9] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese. Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator. In *Proceedings of the ACM SIGCOMM Conference*, pages 255–266. ACM, 2009.

[10] M. Lee, N. Duffield, and R. R. Kompella. Not all microseconds are equal: Fine-grained per-flow measurements with reference latency interpolation. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 27–38. ACM, 2010.

[11] M. Lee, N. Duffield, and R. R. Kompella. Maple: A scalable architecture for maintaining packet latency measurements. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, pages 101–114. ACM, 2012.

[12] M. Lee, S. Goldberg, R. R. Kompella, and G. Varghese. Fine-grained latency and loss measurements in the presence of reordering. *SIGMETRICS Perform. Eval. Rev.*, 39(1):289–300, June 2011.

[13] P. Menage. Cgroups, [online]https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt.

[14] R. Mutia, N. Sadeque, J. Andersson, and A. Johnsson. Time-stamping accuracy in virtualized environments. In *13th International Conference on Advanced Communication Technology*, pages 475–480, Feb 2011.

[15] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *Proceedings of the 12th USENIX NSDI Conference*, pages 117–130, 2015.

[16] P. Phaal. sflow version 5, july 2004, [online]http://sflow.org/sflow_version_5.txt.

[17] K. Phemius and M. Bouet. Monitoring latency with openflow. In *Network and Service Management (CNSM), 2013 9th International Conference on*, pages 122–125, Oct 2013.

[18] M. Shahzad and A. X. Liu. Noise can help: Accurate and efficient per-flow latency measurement without packet probing and time stamping. In *the ACM SIGMETRICS Conference*, pages 207–219. ACM, 2014.

[19] Soltesz, Stephen and Pötzl, Herbert and Fiuczynski, Marc E. and Bavier, Andy and Peterson, Larry. Container-based operating system virtualization. *ACM SIGOPS Operating Systems Review*, 41:275, 2007.

[20] I. Song. IEEE 1588 and PTP, [online]http://events.linuxfoundation.org/sites/events/files/slides/elc_insop_2015.pdf.

[21] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. An updated performance comparison of virtual machines and linux containers. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015.

[22] J. Wang, S. Lian, W. Dong, Y. Liu, and X.-Y. Li. Every packet counts: Fine-grained delay and loss measurement with reordering. In *Proceedings of the IEEE 22nd International Conference on Network Protocols (ICNP)*, pages 95–106. IEEE, 2014.

[23] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha. Flowsense: Monitoring network utilization with zero measurement cost. In *Proceedings of the 14th International Conference on Passive and Active Measurement*, pages 31–41, 2013.

[24] J. Zhang and A. Moore. Traffic trace artifacts due to monitoring via port mirroring. In *Workshop on End-to-End Monitoring Techniques and Services*, pages 1–8, Yearly 2007.