ACTIVE MEASUREMENT OF THE LATENCY IN CLOUD-BASED NETWORKS

Adrian TAUT Iustin A. IVANCIU Eduard LUCHIAN Virgil DOBROTA
Technical University of Cluj-Napoca, Communications Department, 400027 Cluj-Napoca, Romania,
Phone: +40-264-401226, Fax: +40-264-401916,
Emails: adriantaut@gmail.com, {Iustin.Ivanciu, Eduard.Luchian, Virgil.Dobrota}@com.utcluj.ro

Abstract: This paper presents a software tool for the active measurement of the one-way delay (latency) based on cyclic-path delay method in cloud. The principle consists in sending packets from the source node to every node belonging to a multicast group. Copies of the original packet will be forwarded through every possible path in the network before returning to the source node. The original idea is represented by the breaking of the well-known flooding rules which do not allow sending back the packet to the issuing node. The software tool is composed of two modules: one in C for measuring the latency on cyclic paths (packet generation and timestamping) and one in MATLAB for the estimation of one-way delays. The equation system derived from the cyclic-path delays is underdetermined, therefore the estimation of some latencies is needed. The estimation problem is formulated as a constrained optimization problem. The calibration was performed by means of a software tool, previously developed within Unified Communications Labs (UC Labs), where all the authors are currently working. The values of the latencies have a precision of nanoseconds and a good tracking ability of unexpected delay variations. The tool is scalable for both clouds orchestrated by OpenStack and physical networks, regardless of the topology and number of nodes.

Keywords: *active measurements; cloud; cyclic-path delay; latency.*

I. INTRODUCTION

The one-way delay (OWD) is one of the most interesting characteristics of the Internet due to its unpredictable variations. OWD offers information related to network topology, congestion state or route changes. The four main components of OWD are processing delay, transmission delay, propagation delay and queueing delay.

In the literature we can find many solutions for measuring OWD using NTP (Network Time Protocol) as the synchronization protocol. NTP is a protocol used to synchronize the clock of a client to a reference time source [10], always consisting of a hierarchy of primary and secondary time servers [15]. Depending on the location of NTP servers, three OWD measurement setups are provided in [16]. Setup I uses two NTP servers located in the same network as the sender and receiver with an RTD less than 1 ms, thus the uncertainty of the OWD is below ±500μs. Setup II makes use of one common NTP server located in the external GEANT network [5]. The uncertainty of the OWD measurement is still far below ±1 ms even if the synchronization conditions were much worse compared to the previous setup. In Setup III one common NTP server located in Ireland was used and the influence of asymmetric routing has been tested. The traffic in one direction is routed through the Pan-European GEANT network (20 ms OWD) and the traffic in opposite direction is routed via the Telia network (37 ms OWD), resulting an average error of 8 ms, which is about one half of the OWD difference in both directions. The authors of [2] describe a hardware solution capable of inserting a timestamp in the packets just before sending them. This solution gives a higher accuracy in terms of OWD measurements (error less than 100 ns).

One of the most accurate methods of clock synchronization is based on attaching a Global Positioning System (GPS) receiver that synchronizes to the atomic clocks of GPS satellites [17]. However, it requires additional hardware systems such as antenna and distribution equipment for every group of hosts, which make its use impractical in the viewpoint of economy and convenience. Many architectures using the GPS system for measuring OWD can be found in literature. In [12] the authors propose an active measurement architecture which uses DAG boards [3] to obtain timestamps. These boards have their own processor and PCI interface; they are synchronized to a GPS receiver and achieve an order of precision lower than 100 ns.

Another GPS-based architecture is described in [11]. It is composed of several measurement points, a measurement system and a data collector. Based on the packet`s timestamp and packet ID delivered to the data collector, the OWDs are calculated and stored in a local database. Paper [1] tries to achieve a better trade-off between cost and accuracy of the OWD measurement system with respect to the application accuracy requirements. The target is an accuracy of tens of μs in end-to-end communications with typical delays in the millisecond range.

The third category of OWD measurement systems are related with the IEEE 1588 synchronization method. The authors of [7] designed two architectures for WLAN networks. The first one is implemented using a Linux PC platform and a standard IEEE 802.11 WLAN providing a 660 ns clock offset. The second prototype uses an

Manuscript received February 18, 2017; revised March 19, 2017

embedded processor which performs hardware timestamping and achieves a three decades better synchronization (1.1 ns clock offset) compared to the first solution. A comparison of the three synchronization methods presented before can be found in Table 1.

Paper [14] demonstrates the IEEE 1588 capability to synchronize the internal clocks of network equipment to within 10 μs. In [9] a software based master selection mechanism is proposed with performances up to 1 μs. Paper [4] improves the IEEE 1588 synchronization technique by dealing with the sources of timing fluctuation. The main sources of timing errors are repeaters, switches, routers and oscillators within a node.

Table 1. Accuracy of time synchronization methods

Protocol	Characteristics	Interface	Synchronization accuracy
GPS	Standard Output from GPS receivers	Directly wired	1 μs
NTP	IPS (Internal Protocol Suite) standard for time synchronization	Ethernet	WAN: 10-20 ms LAN: <1 ms
IEEE 1588	Standard for instrumentation devices	Ethernet	< 1 μs

OWAMP is an IPv4/IPv6 command line client application and a policy daemon used to determine one way delay between hosts. It is an implementation of the OWAMP protocol as defined in [20]. For meaningful measurements the clocks must be synchronized and stable (the usage of virtualization technologies is not recommended), and also the power-management features of most PC hardware should be disabled, avoiding clock instability caused by speeding up and slowing down the processor. In [18], the authors describe the OWAMP measuring tool as being composed of two inter-dependent protocols, the OWAMP-Control and the OWAMP-Test, which can ensure a complete distinction between the two type of entities in the system: client and server. The OWAMP-Control protocol runs over TCP and is used to initiate and manage measurement sessions and to receive their results. At the beginning of each session, there is a negotiation about the sender and receiver addresses, the port numbers, the time session starts and its duration, the packet size and the mean interval between two consecutive packets. The OWAMP-Test runs over UDP and is used to exchange test packets between sender and receiver. These packets include a Timestamp field that contains the time instant of packet emission. Besides, each packet also indicates if the sender is synchronized with some exterior system (using GPS or NTP) and includes a Sequence Number.

The one-way delay measurement literature also presents J-OWAMP, a Java implementation of the One-Way Active Measurement Protocol (OWAMP). The tests performed in [21] show that the obtained results for both implementations of OWAMP (Internet2 OWAMP and J-OWAMP) could interoperate and achieve consistent results. Same as the previous implementation of OWAMP Protocol, J-OWAMP requires synchronized and stable clocks.

Rude/Crude is a command line traffic generator and measurement tool for UDP, offering only IPv4 support. During the measurement tests, both the transmitter (RUDE) and the receiver (CRUDE) must have synchronized clocks in order to have meaningful results of the one-way delay. RUDE stands for Real-time UDP Data Emitter and CRUDE for Collector for RUDE. The latter is a receiver and logging utility for packets generated by RUDE. It generates a snapshot about each received packet. The snapshot includes a stream identifier specified in the RUDE configuration, sequence number of the packet within the stream, transmitting timestamp, receiving timestamp and the packet size in bytes. The snapshots can be either displayed on-the-fly in a text form on the standard output or logged in a binary form to the file to be decoded into the text form later. The timestamps indicating when each packet has been sent and received are stored with 1 microsecond resolution and allow precise measurements of the parameters mentioned in Table 2.

Table 2. Latency measurement tools

OWD Measure-ment Tool	Operating System	Network Protocol	Trans-port Protocol	Measured Parameters
OWAMP (J-OWAMP)	Linux, Windows	IPv4, IPv6	TCP, UDP	Throughput, packet loss, one-way delay, jitter
RUDE/ CRUDE	Linux, Solaris, SunOS, FreeBSD	IPv4	UDP	Throughput, packet loss, one-way delay, jitter

Only a few studies have evaluated the end-to-end network performances of cloud services. Paper [19] studies the performance of Amazon EC2 cloud computing system. The results show that even in lightly utilization of the data center, virtualization can still cause abnormal delay variations. The experiment setup is composed of 750 small Amazon EC2 instance pairs and 150 medium Amazon EC2 instance pairs and 5000 ping probes ending with the conclusion that delay variations can be a hundred times larger than the propagation delay between two end hosts.

In [13] the authors try to find a correlation between the changes in delay and other QoS parameters of global distributed cloud computing applications by performing measurements done throughout 24 continuous hours between different network points. The latency and throughput are varying with some rather dramatic changes without having any explanation. Due to the fact that all the measurements in cloud-based networks are ping-based and have major variations we cannot discuss about accuracy regarding the OWD measurement.

Our work addresses the drawbacks of the existing solutions and it proposes a scalable cloud-based software tool using the cyclic-path delay method. Note that the topology in a cloud may not be predefined and the large number of nodes may change their activation status very often.

The rest of the paper is organized as follows. Section II describes the cyclic-path delay measurement, followed by implementation in Section III. The experimental results in Section IV refer to a testbed orchestrated by OpenStack. Conclusions and future work end the paper.

II. CYCLIC-PATH DELAY MEASUREMENTS

In order to perform the measurements, a source node sends a probe packet that is forwarded along several nodes until the packet returns to the source node. The time the packet is processed in the intermediate nodes can be computed and subtracted from the total cyclic-path time.

For each link i→j in Ɛ, let $x_{i,j}$ be the one-way delay from i to j on that link. Let $\hat{x}_{i,j}$ be the estimate of $x_{i,j}$. The estimation problem is described in [6] and is formulated as a constrained optimization problem, where the variables are $\vec{x} = \{x_{i,j}\}$ and the constraints are the measured cyclic-path delays. Assume that L measurements are taken. Let $a_{l\{i,j\}} = 1$ if link {i,j} appears along the path of the *l*-th measurement. Let $\propto_l$ be the measured cyclic-path delay in the *l*-th measurement. The measurement constraints are given by $A \cdot \vec{x} = \vec{\alpha}$, where A is the matrix whose elements are $\{a_{l\{i,j\}}\}$ and $\vec{\alpha}$ is a vector whose elements are $\{\propto_l\}$. The aim is to determine $\hat{\vec{x}}$ that yields the next least square error criteria:

$$\mathbf{min}\{ \int_{\Omega} |\vec{x} - \hat{\vec{x}}|^2 \, d\vec{x} \} \tag{1}$$

under the constraints:

$$\Omega = \{\vec{x} \mid \boldsymbol{x_{ij} > 0}; \boldsymbol{A \cdot \vec{x} = \vec{\alpha}}\} \tag{2}$$

Any further information about $x_{i,j}$ can be used as additional constraints in the definition of Ω.

1. Analysis

Since the estimation of the one-way delay is based on measuring cyclic-path delays, the authors in [6] focus on how many such measurements can and should be taken. Apparently, the more measurements are taken, the better, but in this case the total number of cyclic-paths would be huge. It is shown that in an *N-node* fully connected network the total number of cyclic-paths that start at a specific node and pass through each intermediate node once is:

$$\sum_{i=2}^{N-1} \frac{(N-1)!}{(N-i)!} \tag{3}$$

and the total number of cyclic-paths starting at any node is:

$$N \cdot \sum_{i=2}^{N-1} \frac{(N-1)!}{(N-i)!} > N \cdot (N-1)!, \forall\, N > 2 \tag{4}$$

The number of variables (delays on each link) is $N \cdot (N-1)$ in a fully connected network, so there are many more cyclic-path measurements than there are variables. This leads to the conclusion that from the whole set of cyclic-paths, only a part of them are independent[8]. If there were $N \cdot (N-1)$ independent cyclic-paths, then the system of equations obtained from the cyclic-path measurements would have been a determined one, so the one-way delays of each link could have been obtained.

2. Numerical Solution Method

A numerical procedure to simplify the computation developed based on approximating the integral by a sum over all the vectors $\vec{x}$ with the constraint $x_{ij} \geq 0$, then solve the equation $A \cdot \vec{x} = \vec{\alpha}$. Depending on the running time the resolution can be as fine as desired. Since the maximal number of independent equations that could be derived is $E - (N-1)$, $N-1$ independent equations $(\omega_1 = \beta_1, \omega_2 = \beta_2, \dots, \omega_{N-1} = \beta_{N-1})$ must be added, where ω`s are part of the variables x_{ij}. In this way a set of E equations with E variables written in matrix form as $B \cdot \vec{x} = \vec{\eta}$ is obtained. The $E \times E$ matrix B is:

$$B = \begin{pmatrix} & & & & A & & & \\ 1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ & & & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{5}$$

with the diagonal representing the extra $N-1$ equations $(\omega_1 = \beta_1, \omega_2 = \beta_2, \dots, \omega_{N-1} = \beta_{N-1})$.

Also the $\vec{\eta}$ vector is exemplified bellow, being composed in the first part of the measured cyclic-path delays corresponding to the $E - (N-1)$ independent equations, followed by the $(\beta_1, \beta_2, \dots, \beta_{N-1})$ which will be further derived by applying the appropriate constraints to the already derived set of equations.

$$\vec{\eta} = \begin{pmatrix} \propto_1 \\ \vdots \\ \propto_{N-1} \\ \beta_1 \\ \vdots \\ \beta_{N-1} \end{pmatrix} \tag{6}$$

Due to the fact that all equations are independent, B is a non-singular matrix, so there exists B^{-1} and the vector $\vec{x}$ can be computed. The next step is to choose $(\beta_1, \beta_2, \dots, \beta_{N-1})$ such that all elements of $\vec{\beta}$ to be positive, derive the vector $\vec{x}$ and check if all elements are non-negative $(x_{ij} \geq 0, \forall x_{ij})$. Another constraint can be derived from the fact that the ω`s are part of the x_{ij}, hence each of them must fulfil an equation of the form $\omega_i + Delay\{\cdot\} = \alpha_l, \forall l$. Furthermore, $\vec{\beta}$ can be restricted only to $\vec{\beta}$ where $\beta_i \leq \alpha_l$.

Now that $\vec{\beta}$ is properly chosen we can solve the equation:

$$\vec{x} = \beta^{-1} \cdot \vec{\eta} \tag{7}$$

and search for all non-negative x_{ij} to find the entire set of $\vec{x} \in \Omega$. The last step is to compute $\hat{\vec{x}}$:

$$\hat{\vec{x}} = \frac{1}{m} \sum_{\vec{x}_r \in \Omega} \vec{x}_r \tag{8}$$

where m is the number of points in Ω.

III. IMPLEMENTATION

This section presents a solution for the estimation of one-way delays from cyclic-path delay measurements. The cyclic-path delays measurements are performed using a source node that will forward a packet in the network which will be multiplied and will pass through all nodes in the network. Finally the packet will return to the source node, where the cyclic-path delays will be measured by subtracting from the arrival time the starting time. These

measurements are then expressed in terms of one-way delay variables. The resulting equation system is underdetermined, so an estimation of one-way delay is performed in order to output a final solution of one-way delays on each link of a network.

The algorithm presented in Section II was implemented in two main steps: cyclic-path delay measurements (C-based application) and derivation of one-way delays (MATLAB). The programs are providing continuous estimations of OWDs on the network links that can be further used in routing algorithms or to monitor network performances. Both the C and MATLAB programs are designed to be topology independent and need no a priori configuration in order to run.

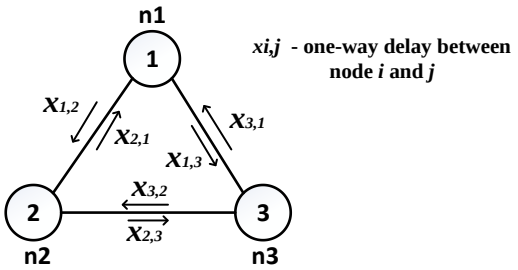

Figure 1. Illustration of network diagram and one-way delays

A simple example is presented in Figure 1. The source node n1 starts the process by forwarding a so-called multicast probe packet in the network and keeps the starting time in a *struct timeval startProgram* variable. All the nodes interested in receiving traffic from the multicast group `226.1.1.1` will concatenate the received packet with their hostname and the processing time in each specific node, then will flood the newly created packet on all network interfaces, including to the one on which the packet was received, breaking the flooding rules. The multiplied packets will pass through all the nodes and will finally return to the source node *n1*, which only collects information without forwarding anymore packets. Without any kind of synchronization, *n1* will measure the cyclic-path delay for each incoming packet by subtracting from the receiving time the starting time with a precision of nanoseconds. All the information is written in a *delay.txt* file that will be further processed by the MATLAB program.

Since we are interested only in the one-way delays on the network links, the processing time in each of the nodes the packets are passing through should be subtracted from the cyclic-path delay. In order to do this the incoming packets are timestamped when they reach the Unix Kernel by using the SO_TIMESTAMPNS socket option with a nanoseconds resolution[8]. The *setsockopt()* receives as parameters the software timestamping option SO_TIMESTAMPNS and an *int enabled =1* variable which sets the SO_TIMESTAMPNS option to *TRUE*. The message diagram of the whole measurement process is illustrated using Figure 2.

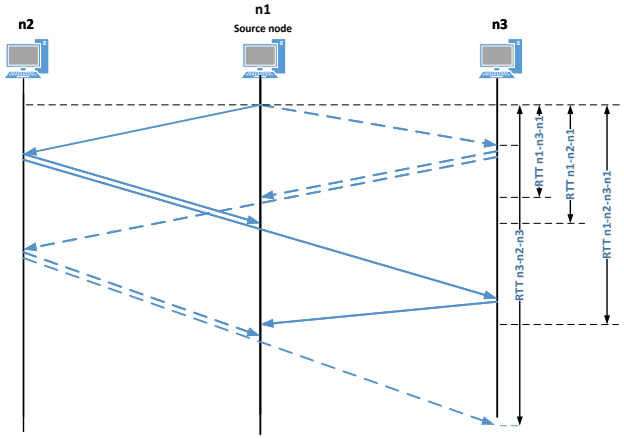

Figure 2. Message diagram in the cyclic-path delay measurement process

The second part of the implementation consists of a software program written in MATLAB whose purpose is to parse the *delay.txt* file, to extract the $E-(N-1)$, to add the extra $(N-1)$ equations and finally to output the OWDs on each directed link of the tested network. Since the cyclic-path delay measurements are performed only on the source node, only there should be installed MATLAB.

IV. EXPERIMENTAL RESULTS

The implemented solution will be deployed and tested on several network topologies created in two private OpenStack cloud architectures. A detailed analysis of the results and a comparison with an existing OWD measurement tool will be performed to demonstrate its correct functionality.

1. *Fully Connected 3-Node Network Deployed in Compute Node 2*

The OpenStack cloud architecture of this testbed is composed of a cloud controller and two compute nodes. Three CentOS 7 VMs were created in Compute Node 2 with the following specifications: a) Virtual Machine *n1* used as source node – has the role of starting the measurement process, collecting and processing data; b) Virtual Machine *n2, n3* – have the role of receiving and forwarding packets. VMs *n2* and *n3* have m1.small flavor of OpenStack, this means that they are running on 2GB RAM, 1 VCPU and 20GB HDD each. VM *n1* has m1.medium flavor, so it is running on 4GB RAM, 2 VCPU and 40GB HDD.

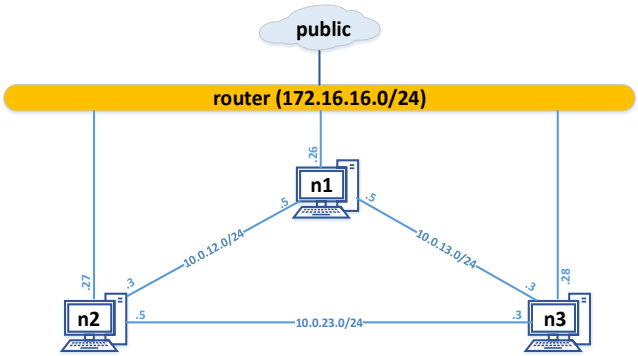

Figure 3. Fully connected 3-node network deployed in Compute Node 2

By using the Horizon interface of OpenStack the network topology can be displayed (Figure 3). All VMs are connected on eth0 to a virtual router in the network 172.16.16.0. The router is connected to the public network and gives the VMs access to Internet for downloading installation packages. The other network interfaces are used to build the fully connected 3-node network.

2. Fully Connected 3-Node Network Deployed on Two Compute Nodes

The same network topology as the one described before is used here too, but this time the node *n3* is launched in Compute Node 1. By using the Horizon interface of OpenStack it can be seen that the network topology is not influenced by the fact that the VMs are running in different Compute Nodes (Figure 4).

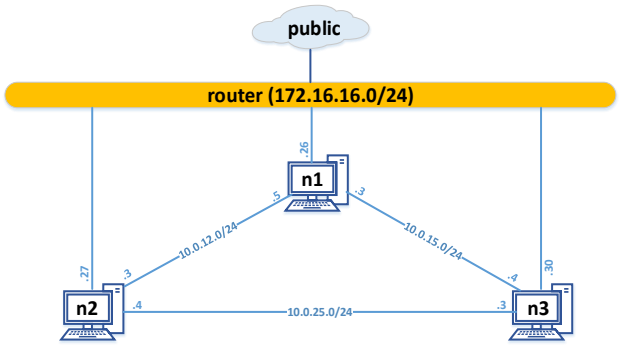

Figure 4. Fully connected 3-node network deployed in two Compute Nodes

3. Comparison with RUDE/ CRUDE

In order to test the performance of the estimation tool, a comparison with the reference tool RUDE/CRUDE was performed. As mentioned earlier, RUDE/CRUDE requires precise synchronization between the network nodes. However, the best we could do was to provide NTP synchronization. Moreover, RUDE/ CRUDE only allows measuring OWD between two nodes at a time, while the tool presented in this paper is scalable to higher networks. While this comparison is far from perfect, it gives us a basic idea on the order of magnitude of the expected results. The output of RUDE/CRUDE is given in the snippet below:

```
[root@scilab rude]# crude -s 0030
Flow_ID=30
Packets: received=101   out-of-seq=0
lost(est)=0
Total bytes received=136855
Sequence numbers: first=0   last=100
Delay: average = 0.002310   jitter=0.000020
seconds
Absolute maximum jitter=0.000535   seconds
Throughput=68450.1   Bps (from first to last
packet received)

crude: captured/processed 101 packets
[root@scilab rude]# crude -s 0030
Runtime statistics results:
---------------------------

Flow_ID=30
Packets: received=101   out-of-seq=0
lost(est)=0
Total bytes received=136855
Sequence numbers: first=0   last=100
Delay: average = 0.002098   jitter=0.000018
seconds
Absolute maximum jitter=0.000427   seconds
Throughput=68451.7   Bps (from first to last
packet received)

crude: captured/processed 101 packets
```

The results presented above show that the average OWD measured by RUDE/CRUDE has the order of magnitude of milliseconds. In this particular case, the OWD values are around 2 ms. A simple ping test however shows that the RTT between these two nodes is less than 1ms. We believe that these differences are caused by the poor synchronization accuracy provided by NTP. The next experiment was performed using our estimation tool. It runs for 200 seconds on the network topology with 3-nodes. There is no other traffic on the network except the small amount of traffic generated by the *multicastOWD.c* program.

Figures 5-10 shows that the results obtained with the *multicastOWD* tool are in the order of hundreds of microseconds, comparable to the results provided by ICMP, far less than the ones obtained with RUDE/CRUDE.

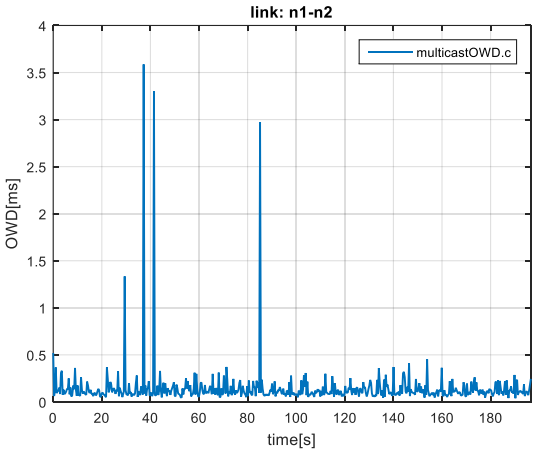

Figure 5. Estimation of OWD on the network link n1-n2 for a fully connected 3-node network

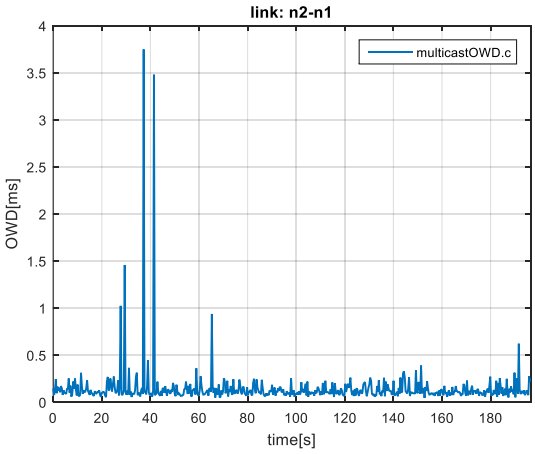

Figure 6. Estimation of OWD on the network link n2-n1 for a fully connected 3-node network

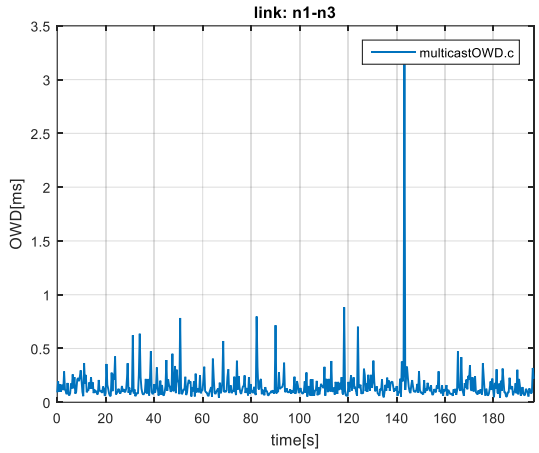

Figure 7. Estimation of OWD on the network link n1-n3 for a fully connected 3-node network

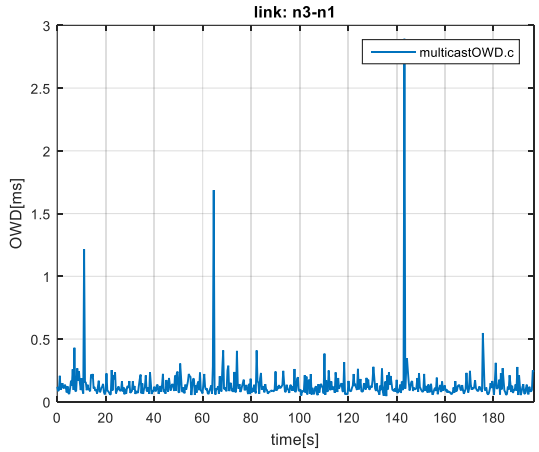

Figure 8. Estimation of OWD on the network link n3-n1 for a fully connected 3-node network

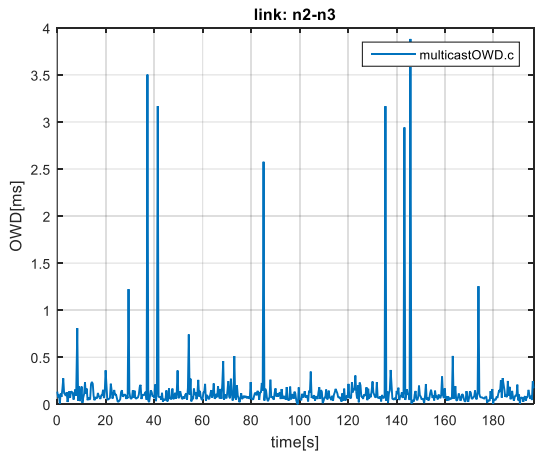

Figure 9. Estimation of OWD on the network link n2-n3 for a fully connected 3-node network

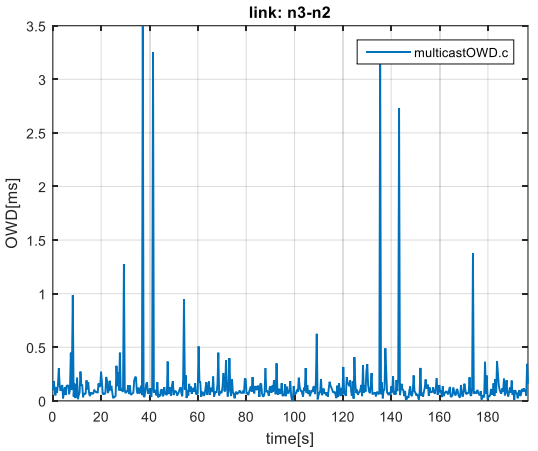

Figure 10. Estimation of OWDs on the network link n3-n2 for a fully connected 3-node network

An interesting observation reffers to the peak values that are almost constant in value on the majority of the network links. There are high peaks around 40 seconds that are present on 4 of the 6 links of the network (*n1→n2, n2→ n1, n2→ n3, n3→n2).* In the rest of the time the network performance looks normal with values of OWDs around hundred microseconds. Moreover it can easily be observed that these peaks occur almost simultaneously on every link, which leads to the conclusion that the prioritizing algorithm of OpenStack is responsible.

4. Comparison with other OWD Measurement Tool

The first experiment aims to compare the results obtained using the proposed estimation method with another OWD measurement tool, previously developed within Unified Communications Laboratories (UCLabs - http://users.utcluj.ro/~uclabs/) [21]. This software tool called *measureEstOWD* performs the active measurement of the round-trip time and estimates the latency as RTT/2. Both tools ran 200 seconds to measure the OWD from *n1→n2* and from *n2→n1*. The outputs of both the *measureEstOWD.c* and *multicastOWD.c* methods are plotted in Figure 11 and Figure 12.

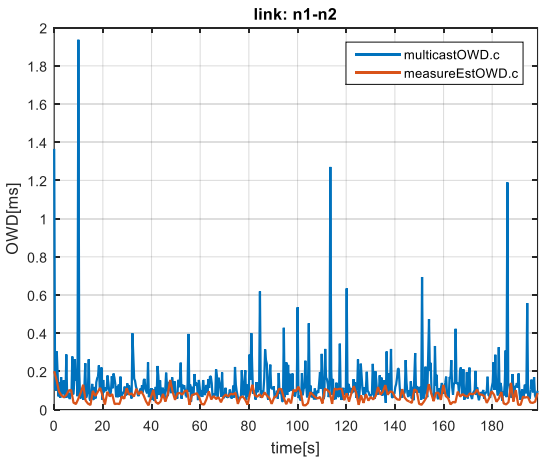

Figure 11. multicastOWD.c and measureEstOWD.c outputs for link n1→ n2

Both solutions provide similar results whenever there is no traffic on the network. The output of *measureEstOWD.c* program performs an average of the OWDs and output results once per second, while the *multicastOWD.c* outputs a set of OWDs once at every 200 milliseconds. The main difference between the two outputs consists in peaks that can be easily observed. This is due to the fact that the *multicastOWD.c* program performs the estimation more often. Thus this increases the possibility of measuring higher latencies due to the OpenStack tendency of prioritizing certain management processes. In opposition, the *measureEstOWD* performs an averaging over a period of 1 second.

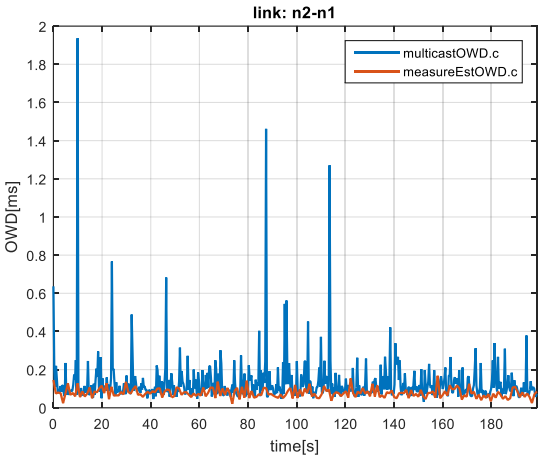

Figure 12. multicastOWD.c and measureEstOWD.c outputs for link n2→n1

5. *Tracking Accuracy of the Estimation Tool*

In this experiment an additional delay of 10 ms from second 60 until second 120 on the network link n2→ n3 is introduced. This test aims to measure the tracking ability of the estimation tool in case of abrupt delay variations. According to Figure 13 the response time of the measurement tool is very small and the variations are detected. Moreover, the estimation method is not prone to errors in case of link asymmetry, which often occurs in real life scenarios.

6. *The Effect of the Measurement Process Duration on the Estimation Accuracy*

Let us investigate now the behavior of the *multicastOWD* program when one of the network links has a delay higher than the timer set for cyclic-path delay measurement process (e.g. 200 ms). However, this only holds supposing that the sum of the OWDs on the longest path will not exceed this value. Otherwise, the OWD on some of the links composing that path will not be estimated at all. This however does not affect the measurement process on the other links. We will set an additional delay of 250 ms on the network interface of node *n2* which is connected to node *n3*. By setting this interval, no packet will be forwarded from *n2→ n3* in the interval of 200 ms, so the OWD of this link can not be measured. Figure 14 shows that link *n2→ n3* does not appear on the graph, but in the same time, it does not influence the measurement and the estimation process.

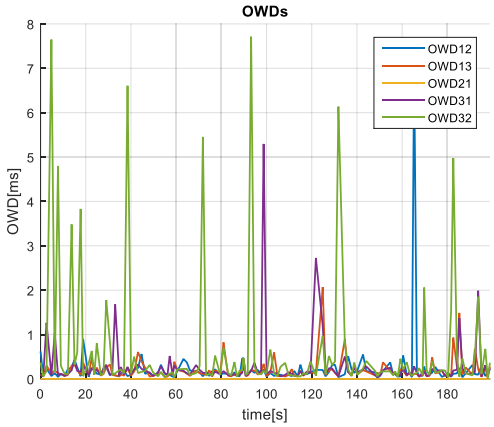

Figure 14. OWDs output with delays higher than program runtime (e.g. 200 ms)

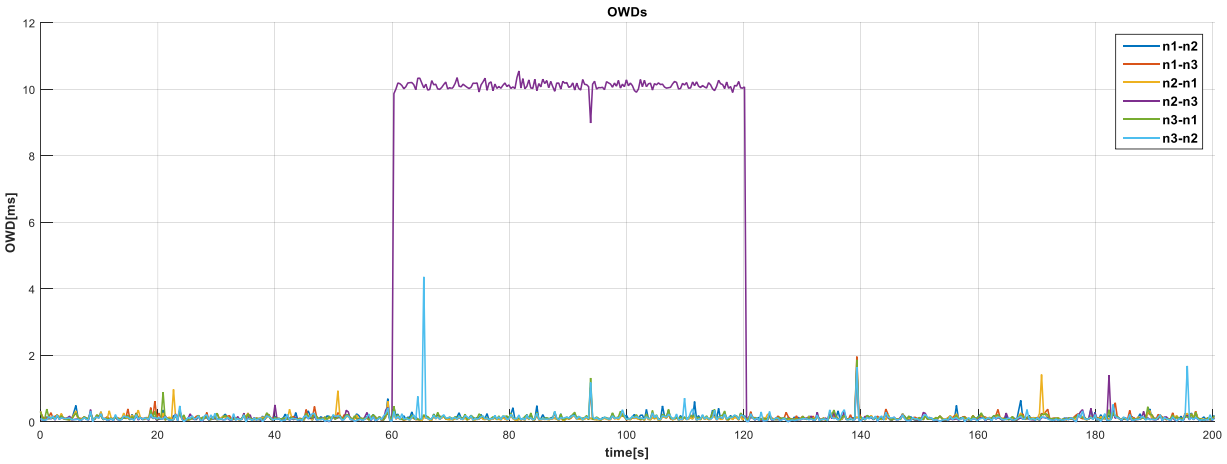

Figure 13. Additional OWD introduced on one of the links

7. Estimating OWDs in a Fully-Connected 3-Node Network Deployed on Two Compute Nodes

In this experiment the OWD estimation tool is deployed in a fully connected 3-node network. The aim of this approach is to split the traffic injected by the cyclic-path delay measurement process between the two Compute Nodes of the OpenStack architecture in UC Labs in order to observe the OWD variations. Compute Node 2 hosts the nodes *n1* and *n2*, while Compute Node 1 hosts node *n3*.

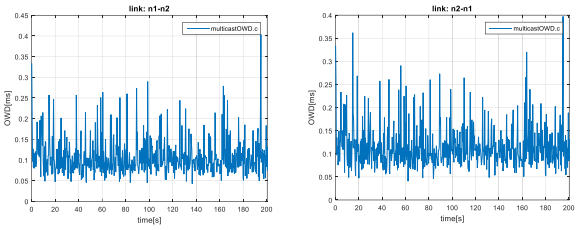

Figure 15. Estimation of OWD for the links n1-n2 and respectively n2-n1

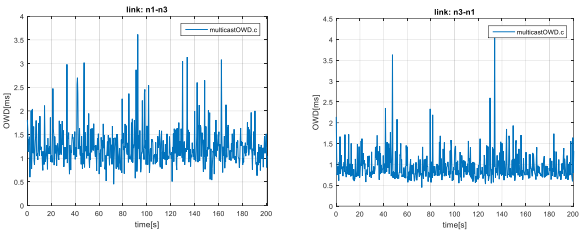

Figure 16. Estimation of OWD for the links n1-n3 and respectively n3-n1

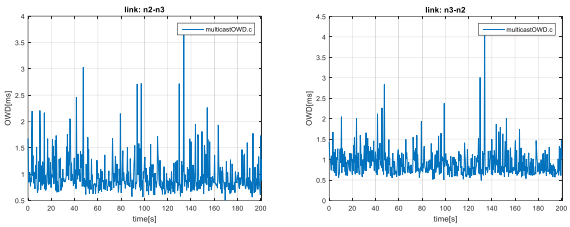

Figure 17. Estimation of OWD for the links n2-n3 and respectively n3-n2

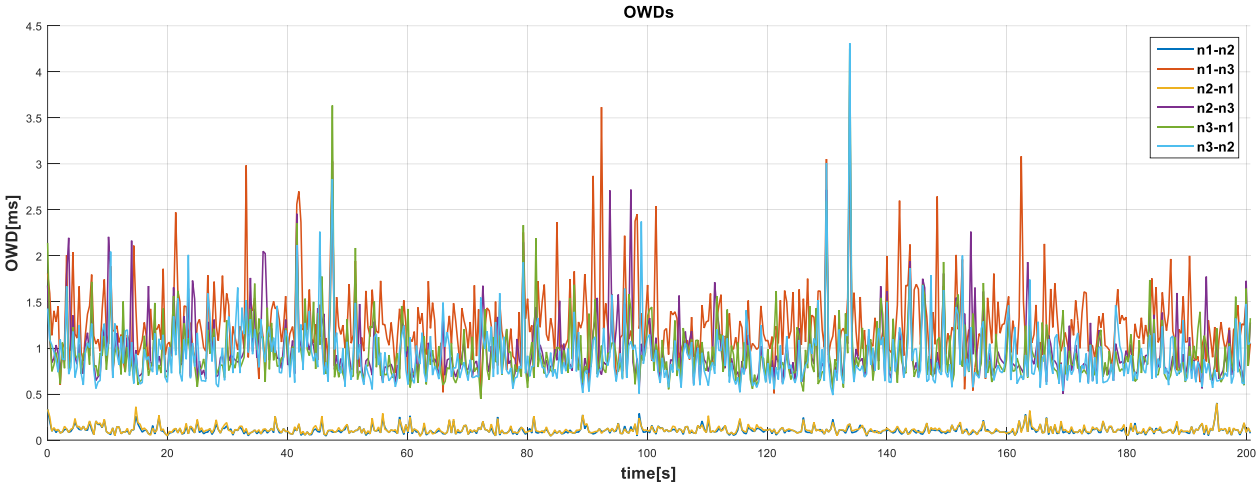

Figure 18. Estimation of all OWDs for a fully connected 3-node network deployed on two Compute Nodes

The OWDs between the nodes *n1* and *n2* are illustrated in Figure 15 and their values are approximately the same as in the previous experiments since both the VMs are running on Compute Node 2. The OWD on the other links is higher due to the fact that the traffic on these links is transmitted using GRE tunnels and it passes through physical network links and interfaces.

The peaks in terms of OWD are not present anymore for the links *n1* → *n2* and *n2* → *n1*, mainly because part of the traffic is forwarded to Compute Node 1. The delays caused by the communication with Compute Node 1 which spreads the traffic in time, together with the computational power of Compute Node 2, seems to be enough for OWDs between nodes *n1* and *n2* without high delays.

The other four directional links which forward traffic between the two Compute Nodes have poorer, but expected network performances in terms of OWD (Figures 16-17). The peaks caused by the hardware characteristics of Compute Node 1 are still present though (Figure 18).

V. CONCLUSIONS AND FUTURE WORK

This paper presented a new implementation of the cyclic-path delay method, suitable for the active measurement of latencies in cloud-based networks, with a precision of nanoseconds and no need for clock synchronization. The solution is scalable due to its principle of being independent of topology and number of nodes, the only requirement being higher computational power for the node responsible with the MATLAB data processing. The tracking ability of the estimation tool in case of unexpected delay variations was also tested. The OWD variations artificially introduced on the network links are immediately detected, hence the network performances in terms of OWD are updated in real time. Future work envisages the deployment of this measurement method in a multi-side cloud testbed.

Furthermore we plan to use this software implementation in conjunction with other QoS tools for routing decisions in software-defined networks.

ACKNOWLEDGMENTS

This work was partially supported by the CHIST-ERA "DIONASYS" project. However the views expressed in this paper are solely those of the authors and do not necessarily represent the views of the entire project.

REFERENCES

[1] D. Adami, R. Garroppo, S. Giordano, and S. Lucetti, "On synchronization techniques: performance and impact on time metrics monitoring", *Int. J. Commun. Syst.*, Vol. 16, No. 4, pp. 273-290, 2003.

[2] D. Constantinescu, P. Carlsson, A. Popescu, and A. A. Nilsson, "Measurement of one-way Internet packet delay", *Proceedings of 17th Nordic Teletraffic Seminar - NTS17*, Oslo, Norway, pp.1-11, August 2004.

[3] "The dag project," September 2010. [Online]. Available: http://dag.cs.waikato.ac.nz//, [Accessed: June, 2016].

[4] J. Eidson, and B. Hamilton, "IEEE-1588 Node Synchronization Improvement by High Stability Oscillators", *Workshop on IEEE-1588, Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, NISTIR 7070 , pp. 102-112, September 2003.

[5] GÉANT Project (GN4-1), Geant.net, 2017 [Online]. Available: http://www.geant.net/, [Accessed: February, 2017].

[6] O. Gurewitz, and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," *Proceedings IEEE INFOCOM 2001 Conference on Computer Communications*, Vol.2, pp. 1038-1044, 2001.

[7] J. Kannisto, T. Vanhatupa, M. Hännikäinen, and T.D. Hämäläinen, "Software and hardware prototypes of the IEEE1588 precision time protocol on wireless LAN", *Proc. of 14th IEEE Workshop on Local and Metropolitan Area Networks LANMAN 2005*, Chania, Greece, pp.1-6, 8-21 Sept. 2005.

[8] Kernel.org, 2016. [Online]. Available: https://www.kernel.org/doc/Documentation/networking/timestamping/timestamping.c. [Accessed: 15- Jun- 2016].

[9] D. Lee, and G. Allan, "A solution for fault-tolerant IEEE-1588", *Workshop on IEEE-1588, Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, NISTIR 7070 , pp. 225-231, September 2003.

[10] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol (Version 4) specification, implementation and analysis, *RFC 5905, IETF*, June 2010.

[11] S. Niccolini, M. Molina, F. Raspall, and S. Tartarelli, "Design and implementation of a One Way Delay passive measurement system", *Proc. of Network Operations and Management Symposium*, Seoul, South Korea, Vol.1, pp. 469-482, April 2004.

[12] A. Pásztor, and D. Veitch, "A precision infrastructure for active probing", *Citeseer*, 2001.

[13] J. M. Pedersen, M.T. Riaz, J.C. Junior, B. Dubalski, D. Ledzinski, and A. Patel, "Assessing measurements of QoS for global cloud computing services", *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing DASC 2011*, pp. 682-689, 2011.

[14] M. Rowe, "IEEE 1588 keeps time in sync", Test & Measurement World, 2005, [Online]. Available: http://www.reedelectronics.com/tmworld/article/CA529830?text=synthetic+instruments, [Accessed: February, 2017].

[15] A.S. Sethi, H. Gao, and D. Mills, "Management of the Network Time Protocol (NTP) with SNMP*", Computer and Information Sciences Report*, 98-09, 1997.

[16] V. Smotlacha, "One-way delay measurement using NTP synchronization", *Proc. TERENA Networking Conf. 2013*, pp. 1-18, 2013.

[17] K. Stangherlin, W. Lautenschläger, V. Guadagnin, L. Balbinot, R. Balbinot, and V. Roesler, "One-way delay measurement in wired and wireless mobile full-mesh networks", *IEEE Wireless Communications and Networking Conference WCNC 2011*, pp. 1044-1049, March 2011.

[18] H. Veiga, R. Valadas, P. Salvador, A. Nogueira, T. Pfeiffenberger, and F. Strohmeier, "OWAMP Performance and Interoperability Tests", *4th International Workshop on Internet Performance, Simulation, Monitoring and Measurement IPS-MoMe 2006*, Salzburg, pp.1-10, 2006.

[19] G. Wang, and T. E. Ng, *"*The impact of virtualization on network performance of amazon ec2 data center", *Proc. of IEEE INFOCOM 2010*, pp. 1-9, 2010.

[20] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", *RFC 4656*, IETF 2006.

[21] M.V. Ulinic, A.B. Rus, and V. Dobrota, "OpenFlow-Based Implementation of a Gearbox-Like Routing Algorithm Selection in Runtime", *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, ISSN 1221-6542, Vol.55, No.2, pp.23-32, 2014.